DOCUMENT RESUME

ED 454 826	IR 020 805
TITLE	ICCE/ICCAI 2000 Full & Short Papers (Intelligent Tutoring Systems).
PUB DATE	2000-11-00
NOTE	202p.; In: Learning Societies in the New Millennium: Creativity, Caring & Commitments. International Conference on Computers in Education/International Conference on Computer-Assisted Instruction (Taipei, Taiwan, November 21-24, 2000); see IR 020 792. Some figures contain very small and illegible type.
PUB TYPE	Collected Works - General (020) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC09 Plus Postage.
DESCRIPTORS	*Educational Technology; Elementary Secondary Education; *Intelligent Tutoring Systems; Internet; Postsecondary Education; Programmed Tutoring
IDENTIFIERS	Learning Environments; *Web Based Instruction

ABSTRACT

This document contains the full and short papers on intelligent tutoring systems (ITS) from ICCE/ICCAI 2000 (International Conference on Computers in Education/International Conference on Computer-Assisted Instruction) covering the following topics: a framework for Internet-based distributed learning; a fuzzy-based assessment for the Perl tutoring system, a learning environment for problem posing in simple arithmetical word problems; a method of creating counterexamples by using error-based simulation; adaptive programming language tutoring system on the World Wide Web; an agent-based collaborative learning environment for ITS; an agent-based ITS; an educational system that can visualize the behavior of programs on the domain world; an environment for learning by design in the case of learning of search algorithms; an online ITS for elementary algebra; automatic background knowledge construction using genetic algorithms; AWETS (Automatic Web-Based English Testing System); case-based evaluating assistant of novice program; CBR-Tutor--a case-based reasoning approach to an Internet agent-based tutoring system; controlling problem progression in adaptive testing; development of an intelligent learning support system with a large knowledge base; educational agents and the social construction of knowledge; generating interactive explanations by using both images and texts for Micro World; intelligent interactive learning environment design issues; towards a meta-knowledge agent--creating the context for thoughtful instructional systems; modeling the tutor using reinforcement learning; monitoring and verifying mathematical proofs formulated in a restricted natural language; a multimedia ITS for context-free grammar; MyEnglishTeacher--a Web system for academic English teaching; navigation script for the Web; organization of the introductory and motivating stage of activity in a computer tutoring system; the application of uncertainty reasoning for an ITS; the design and implementation of an automatic exercise generator with tagged documents based on the intelligence of students (AEGIS); traversing the case graphs: a computer model for developing case-based learning systems; use of abstraction levels in the design of ITS; and using decision networks for adaptive tutoring. (MES)

Reproductions supplied by EDRS are the best that can be made
from the original document.

ED 454 826

ICCE/ICCAI 2000 Full & Short Papers (Intelligent Tutoring System)

PERMISSION TO REPRODUCE AND
DISSEMINATE THIS MATERIAL HAS
BEEN GRANTED BY

G.H. Marks

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

IR020805

BEST COPY AVAILABLE

ICCE/ICCAI 2000

Learning Societies in the New Millennium : Creativity, Caring & Commitments

HOT NEWS
Background
Committees
Call for Papers
Deadlines
Registration
Program
Paper Acceptance
Reviewers
Venue & Travel Info.
Contact Info.
Host & Sponsors
Related Websites
Discussion Board
Proceedings
HOME

Proceedings

Content

▸Full & Short Papers (Intelligent Tutoring System)

A Framework for Internet Based Distributed Learning
A Fuzzy-based Assessment for Perl Tutoring System
A Learning Environment for Problem Posing in Simple Arithmetical Word Problems
A Method of Creating Counterexamples by Using Error-Based Simulation
Adaptive Programming Language Tutoring System on the Web
Agent-based Collaborative Learning Environment for Intelligent Tutoring Systems (ITS)
An Agent-Based Intelligent Tutoring System
An Educational System that can Visualize Behavior of Programs on the Domain World
An Environment for Learning by Design : In the Case of Learning of Search Algorithms
An on-line ITS for elementary algebra
Automatic Background Knowledge Construction Using Genetic Algorithms
AWETS: An Automatic Web-Based English Testing System
Case-Based Evaluating Assistant of Novice Programs
CBR-Tutor: A Case-Based Reasoning Approach to an Internet Agent-Based Tutoring System
Controlling Problem Progression in Adaptive Testing
Development of Intelligent Learning Support System with Large Knowledge Base
Educational Agents and the Social Construction of Knowledge: Some Issues and Implications
Generating interactive explanations by using both images and texts for Micro World
Intelligent Interactive Learning Environment: Design Issues
Meta-Knowledge Agent: Creates the context for thoughful instructional systems.
Modeling the Tutor Using Reinforcement Learning
Monitoring and Verifying Mathematical Proofs Formulated in a Restricted Natural Language
Multimedia Intelligent Tutoring System for Context-Free Grammar
MyEnglishTeacher A WWW System for Academic English Teaching
Navigation Script for the World Wide Web
Organization of the introductory and motivational stage of activity in a computer tutoring system
The Application of Uncertainty Reasoning for an Intelligent Tutoring System

BEST COPY AVAILABLE

The Design and Implementation of Automatic Exercise Generator with Tagged Documents based on the Intelligence of Students:AEGIS
Traversing the Case Graphs. A Computer Model for Developing Case-based Learning Systems
Use of abstraction levels in the design of intelligent tutoring systems
Using Decision Networks for Adaptive Tutoring

▲ HOME

A Framework for Internet-Based Distributed Learning

Andreas Braun* and Andreas G. Harrer**
**Andersen Consulting, Maximilianstr. 35, D-80539 München,* andreas.braun@ac.com
***Technische Universität München, Orleanstr. 34, D-81667 München,* harrer@in.tum.de

Communication technology as well as the communication infrastructure are both changing rapidly. As a consequence, systems that support web-based learning need to be adapted due to changes in technology. This paper describes a model for web-based learning with intelligent tutoring systems (ITS) that allows separation of the concrete communication from the ITSs' implementation. The resulting framework provides a technical solution to distribute any ITS over a network. The ITS SYPROS is used to illustrate how a classical ITS can be extended to a web-based tutoring system with a maximum of code-reuse. The framework may be used freely with any ITS. To accommodate the needs of various ITSs, our model supports several architectures for distributed adaptive tutoring, including the three different models described in [3]: Master-Slave, Communicating Peers and Centralized Architectures. Our main goals are: • •Make the ITS usable for a wide range of users by supporting any web browser on any operating system.• •Offer a simple, extendable and platform independent framework to ease web-based tutoring.• • Provide a solution without royalties.• •Separate the communication technology from the client and server implementation.• •Enable method invocation and parameter passing semantics over the HTTP protocol to virtually support any web browser and users behind firewalls.• •Offer an simple user accounting and user communication functionality.• •Provide a wrapper to connect to an existing ITS. The Java source code is freely available: http://www.in.tum.de/~herzog/sypros.

Keywords: **Web-Based Learning, System Design and Development, Intelligent Tutoring Systems**

1 Introduction

Classical intelligent tutoring systems (ITS) are often platform dependant and not distributed. Modern, distributed intelligent tutoring systems (DITS) provide a more attractive solution with respect to usability and platform independence. Therefore, a modern distributed infrastructure like the internet with communication techniques like CORBA or RMI is suitable. A stable, safe and extendable basis for communication and cooperative work is needed. However, technology in this area is rapidly changing on the one hand. On the other hand communication technologies like CORBA or RMI are (still) not usable with every client, browser or platform and still have several drawbacks which prevent their usability at least for some users: Macintosh users and users with old browsers or behind firewalls/proxies who also want to use secure socket factories, only to name some.

This paper discribes a model and the resulting framework to overcome such problems. We propose to address these problems by providing an API with the semantics of object oriented remote method calls over HTTP and Servlets. Further functionality that is most likely in common for any DITS (such as user accounting and identification, security and administrative functionality) is implemented and encapsulated for ease of use.

In the current version, SYPROS is an ITS in the domain of the synchronization of parallel processes with semaphores [4], a domain of programming problems.

All the typical modules of an ITS [15] like the expert module with different types of cooperating domain experts [13], the instructional module with different tutoring strategies, the student model with cognitive and motivational traits [12], and the interface module with several support facilities, are fully implemented in SYPROS.

The current version system is a classical ITS for single-user mode and is written in C for UNIX systems. The user interface is based on the X Windows system and therefore the ITS is platform dependent. There is no direct support for multiple clients and no accounting, access control or WWW support as it would be needed for a web-based group learning system, which is our ultimate goal [11]. In the current implementation the user interface is divided at function level from the `intelligence' and database functionality, but is linked to one single executable. The proposed model will provide an application interface (API) for the client and server side. The API will encapsulate various ways of communication over a network using an abstract factory pattern [2,10]. Concrete implementations for Java RMI and servlets are provided. This model is designed to be easily extendable by other means of network transportation (e.g., CORBA or even Sockets). It will include conceptional security at an eligible level. Further, various ways of interfacing to an existing ITS on the server side are given (Java native calls to C/C++ and the connectivity to shell scripts). This factory can also be easily extended. Figure 1 shows the distribution of SYPROS. The servlet proxy Server enables connection for old webbrowsers, running not necessarily on the same machine as the `SyprosServer` implementation. Two clients are connected: "**Old Webbrowser**" connects using the servlet proxy, "**New Webbrowser**" can either use servlet communication or RMI[17]/ CORBA[22] (or anything else).

This work covers two more aspects: a security discussion for the provided model with a special focus on security issues for an ITS and a usability discussion for various platforms and webbrowsers.

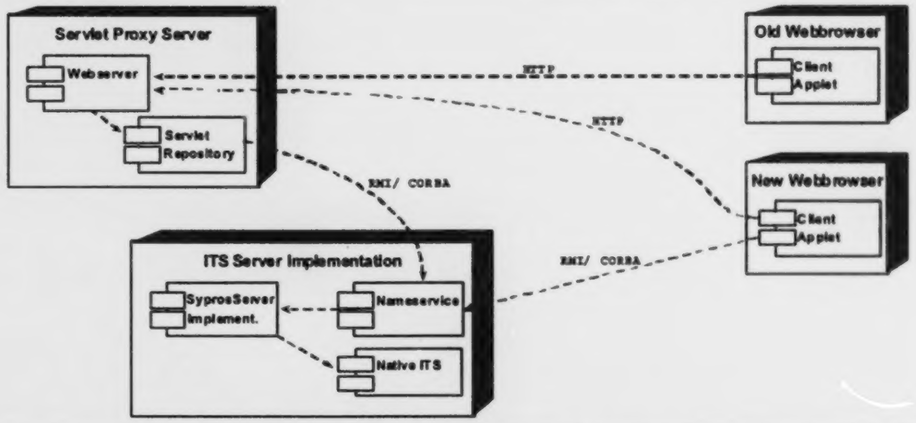


Figure 1. Distribution of client, server and servlet proxy in SYPROS (UML[19]).

Figure 2 shows the different layers for communication and levels of abstraction for a client initiated request. The dotted line between the implementation (application) level and the abstraction denotes that both the client and server implementation are separated from the underlying concrete communication. This model provides transparency in terms of process transparency. (That is, the machine on which the function or method is executed isn't known to the client's application level.) This can be compared to remote procedure calls (RPC) where the client stub and the server skeleton provide a similar transparency. In addition to that, our framework separates the concrete communication (the lowest layer in figure 2) from the application layer using the abstract communication layer. This provides transparency regarding the concrete communication technology used and therefore unburdens the application programmer from changing the application to support new technologies.

For some concrete communication implementations our framework supports language transparency as far as the client's implementation language may differ from the server's (e.g., for CORBA or Servlets).

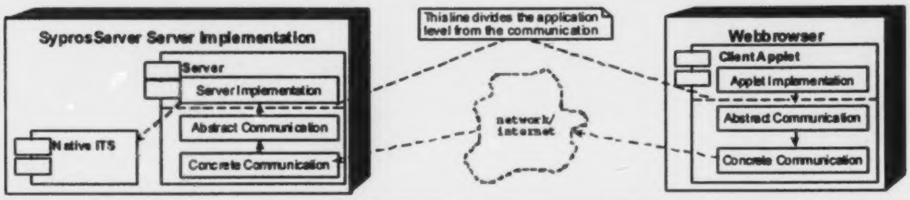

SyprosServer Server Implementation
Server
Server Implementation
Abstract Communication
Concrete Communication
Native ITS
This line divides the application level from the communication
network/ internet
Webbrowser
Client Applet
Applet Implementation
Abstract Communication
Concrete Communication

Figure 2. Levels of abstraction and layers for a request initiated by the client (UML deployment diagram).

2 Requirements

All base functionality for a distributed system is implemented. Remote method invocations are implemented independently from the Java RMI package over a `ComObject` which is JDK 1.1[9] compliant. User accounting, login procedures and access control as well as connection state information is supported directly in the framework.

A wrapper is provided to connect to an existing ITS over Java Native Interface (JNI[14]) or shell script invocation.

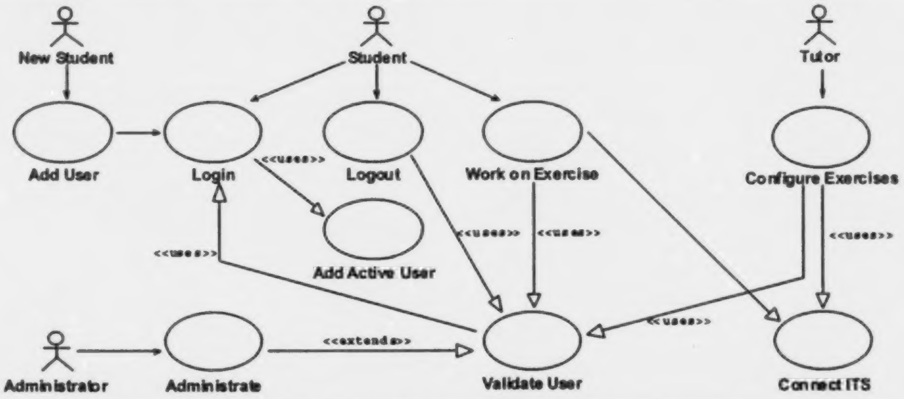

New Student
Student
Tutor
Add User
Login
Logout
Work on Exercise
Configure Exercises
<<uses>>
Add Active User
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<uses>>
Administrator
Administrate
<<extends>>
Validate User
Connect ITS

Figure 3. Some interactions among actors and use cases in SYPROS (UML use case diagram).

The use case diagram in figure 3 shows some of the use cases for SYPROS. Four types of human actors are shown in their interaction with the use cases. "**Student**" denotes an actor who is already known to the system. Therefore, "**Student**" logs into the server by passing the "**Login**" use case. "**Login**" performs authentification for which it `<<uses>>` the "**Validate User**" use case, which has knowledge of all valid user entries and so on. After accepting the user's login request some state information for that connection will be stored "**Add Active User**" and a `UserTicket` object is returned to allow stateful and secure client interaction. (`UserTicket` might be encrypted.)

"**New Student**" is an actor who is not known to the system. (Guests are handled identically.) Therefore, she can create a new user database entry herself ("**Add User**"). Later, the gathered information will be used to log into the system as described before.

An active user ("**Student**") might also use other services on the ITS server side. For example, the "**Work on Exercise**" use case first validates the call against the active users database and then uses "**Connect ITS**" (which interfaces the ITS using the wrapper) to work with the tutoring system.

"**Tutor**" is a human actor who might use the "**Configure Exercise**" use case to set up some exercises or check the student's results. The differing permissions (compared to a student) are handled by the "**Validate User**" use case.

An "**Administrator**" user will not use the client interface to connect to the server in this model. The administrator configures the databases and configuration files. Therefore, "**Administrate**" extends "**Validate User**".

Resulting from the requirements given before our model and the framework should further satisfy the following nonfunctional requirements, pseudo requirements and design goals: The server-side installation should be simple and conceptionally platform independent. It should not be addicted to any specific web server and should work with freely available products such as Apache.

The framework is designed to be fully platform independent using the Java programming language. However, some platform dependencies exist from possible webbrowser incompatibilities and the existing ITS. In order to support old webbrowsers or users behind a firewall or proxy, a servlet repository which acts as a proxy and a servlet based client communication is provided. The communication implementation may be switched online in the client implementation[1].

The SYPROS system can be used by four groups of people: students, guests (users who are not known to the system by now), tutors and administrators (tutors who fulfill administrative functions). Therefore, the framework supports users at different level of permissions (similar to e.g., UNIX or WindowsNT).

The client applet should be small so that it is suitable even for slow modem connections. The classes needed for communication on the client side are less than 20 KB in size (without JCE security). Once the Applet is loaded, the response time of the user interface is short, as it is running locally on the client side.

The response time resulting from the security key generation and secret key exchange (Diffie-Hellman for example) of the Java Cryptography Extension (JCE) is rather long especially for strong keys and due to JCE's implementation in Java (see discussion in section 4).

Performance of the network communication depends on the underlying infrastructure. With most browsers, servlets will have a more overhead than CORBA or RMI.

The communication framework aims to support three possible client-server bindings: Static (the server name is stored in the client application), semi-static (the client locates the server once, e.g., at login time) and dynamic (the client looks up the server each time it needs to connect). Client server binding uses name resolution to find a suitable ITS server in the network. The toolkituses a server string such as "***//{hostname|ip-address}/{service-name}***", just like RMI for either underlying communication infrastructure. At client implementation level, the programmer may decide whether to use static, semi-static or dynamic binding.

Together with the way of client-server binding, stateless and stateful client server connections using tickets are possible. User tickets are invented as "high-level" stateful client-server connection for two reasons: first, the underlying ITS needs to know about the caller; tickets provide an easy way to identify the caller during a learning session. Second, encrypted ticket objects can be used to prevent attacks by intercept and replaying messages (see section 4).

Calling a remote function is somewhat dangerous if the programming language used supports *call-by-reference*[2]. For Java, *call-by-reference* is replaced by a *call-by-copy/restore* semantics. (See Java `RemoteObject` for RMI). A *call-by-copy/restore* semantics can be simulated for servlets using the `EventListener` model. In that case, the servlet proxy uses a `RemoteObject` for the server communication if the communication between the servlet proxy and the server is based on RMI and returns the object to the client using the event model. This may also be encapsulated in the framework.

In case of middleware communication such as CORBA/ RMI the *call-by-copy/restore* semantics can directly rely on the appropriate native semantics. The framework supports synchronous method invocations. Asynchronous calls can be realized using *call-by-copy/ restore*.

---

[1] This feature should be omitted for maximum compatibility with old browsers and Java engines (JDK level 1.1).

[2] Function calls that use *call-by-reference* parameter passing deliver a pointer to the value or data the parameter stores. In a distributed system with different address spaces this triggers side-effects[20].

Java's `try-catch`-statements are used for error handling. Therefore, the framework's error semantics is *at-most-once* by default. At application level, return values might be used to signal unexpected behavior. The SYPROS `login()`-Method for example returns `null` for the `UserTicket` if the server can't accept the login request. Although there are several possible reasons for that (e.g., unknown user, wrong password) their origin is not a communication error.

The resulting framework is described using UML notation for scenarios, use cases and object models[2,19]. The API description is given in standard Java notation[9]. The use of our framework is illustrated by the SYPROS sample.

3 A Model for the Communication Framework

Figure 4 shows the UML diagram for the SYPROS server implementation using the communication framework. The diagram shows two possible extensions for `SyprosServer`: `ComCORBA` and `ComRMI`. In the realworld implementation the programmer has to decide either to use CORBA *or* RMI, as Java does not allow multiple class inheritance.

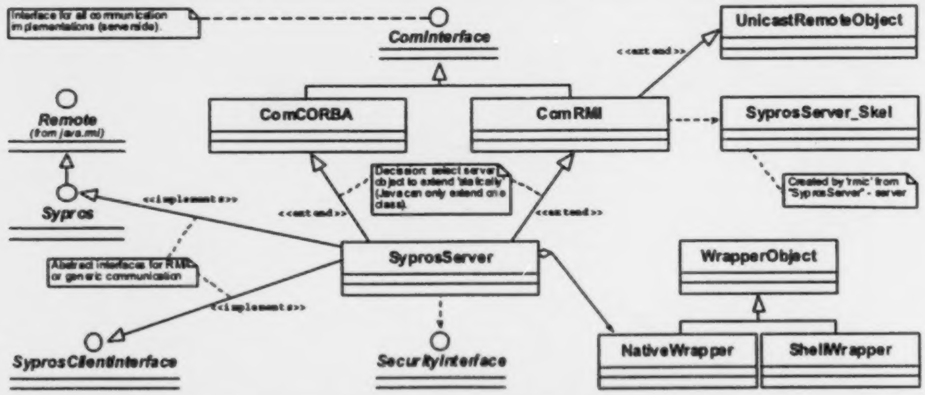

Interface for all communication implementations (serverside).
ComInterface
UnicastRemoteObject
<<extend>>
Remote
(from java.rmi)
ComCORBA
ComRMI
SyprosServer_Skel
Decision: select server object to extend 'statically' (Java can only extend one class).
Created by 'rmic' from "SyprosServer" - server
Sypros
<<implements>>
<<extend>>
<<extend>>
SyprosServer
WrapperObject
Abstract interfaces for RMI or generic communication
<<implements>>
SyprosClientInterface
SecurityInterface
NativeWrapper
ShellWrapper

Figure 4. SYPROS server implementation. Class attributes and methods are omitted. (UML class diagram)

Therefore, there are some specialties in the server implementation: depending on the selected communication technology, the programmer has to change the head of the class definition to extend the right `ComInterface`. Further, the server has to implement the `Sypros` interface which defines the exported functions (for the RMI case). `SyprosClientInterface` contains the same definitions like `Sypros` but doesn't depend on the Java RMI classes. This ensures usability for old webbrowsers (with old Java virtual machines, VM) or clients that don't support RMI for other reasons (Macintosh).

```
import sypros.util.*;
import sypros.com.util.*;
import sypros.com.server.ComRMI;

public class SyprosServer extends} ComRMI implements Sypros {
  public  SyprosServer(String hostName, String servName) throws RemoteException
{
    super(serverHost, servName);  // create bindings
  }
  public static void main(String args[]) {
    setSecurity();                // setup default security
  }
}
```

Figure 5. Client applet implementation for the SYPROS sample.

BEST COPY AVAILABLE

Figure 5 shows the class definition for the SYPROS server implementation using RMI. The underlined statements would have to be changed for a different type of communication technology.

The client implementation allows dynamic switching of the communication technology. Figure 6 shows the UML class diagram for the SYPROS client Applet. As the client communication model uses an abstract factory pattern [2,19] to create the appropriate concrete communication, the client might be a Java Applet or a Java standalone application. (The server could also be connected using the servlet URLs from HTML or other languages.)

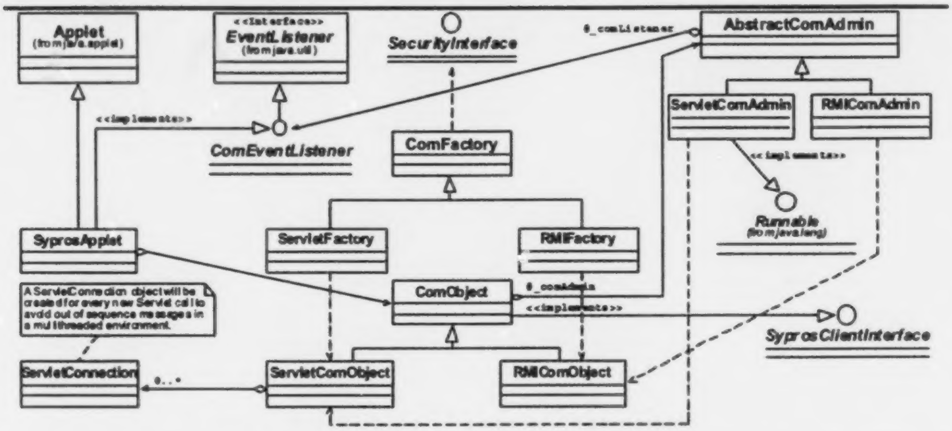

Applet
(from java.applet)
<<Interface>>
EventListener
(from java.util)
SecurityInterface
@_comListener
AbstractComAdmin
<<implements>>
ComEventListener
ComFactory
ServletComAdmin
RMIComAdmin
<<implements>>
Runnable
(from java.lang)
SyprosApplet
ServletFactory
RMIFactory
A ServletConnection object will be created for every new Servlet call to avoid out of sequence messages in a multithreaded environment.
ComObject
@_comAdmin
<<implements>>
SyprosClientInterface
ServletConnection
0..*
ServletComObject
RMIComObject

Figure 6. Client applet implementation for the SYPROS sample (UML class diagram).

The classes in the client model can be seen in three categories. First of all, SyprosApplet is the implementation for the SYPROS client interface. (Plus `Applet`, the parent.) As described before, the implementation needs not to be changed for changing communication technologies.

Then the communication classes themselves: `ComFactory`, `ComObject` and their concrete implementations provide the application interface for the implementation. `ServletConnection` is a helper that provides a per-servlet connection for persistent calls in a multithreaded application.

The `AbstractComAdmin` and its concrete implementations for servlets and RMI currently realize notification for server to client messages using the `EventListener` model and can be used for *call-by-copy/restore* type parameter passing.

4 Conceptual Security

Any internet-based application requires a special focus on security issues. The history of designing secure systems, however, teaches the inadequacy of enhancing existing systems with additional security functionality [8]. To integrate the security functionality for secure web-based tutoring, we included security policies in the framework with a top-down approach. We start by specifying the security requirements as part of the security policy:

Authentification:	All subjects and objects of the system have to be authentificated.
Total access control:	Every access to protected units has to be supervised.
Non repudiation:	Every action performed by a subject can be assigned to it's originator.
Communication privacy:	Dataflow over unsafe networks has to be adequately encrypted.
Availability:	Denial-of-Service attacks should be identified.

To meet the authentification, total access control and no-denial requirements, the framework offers integrated functionality that can be adapted or extended to your application needs. Communication privacy is provided using encrypted transmission (encrypted object serialization) based on the Java Cryptography Extension (JCE). JCE offers secret key agreement protocols (e.g., Diffie-Hellman) and encryption (e.g., Blowfish) with variable key lengths.

Ensuring the availability of a web-based service against denial-of-service attacks is maybe the hardest task. The Servlet-Proxy allows load-balancing, where the typical communication load of an ITS application (little amounts of data, long periods of thinking, infrequent transmissions) can be used to identify attacks.

5 Conclusions and Outlook

Our framework offers an easy and extendable basis for web-based distributed tutoring. The communication technology, security and ITS-integration can be easily adapted to the specific needs of an existing ITS as well as to changing communication or security technologies without rewriting the implementation for the ITS clients or server.

User accounting and access rights deliver the basis to support groups of students. However, support for cooperative work should be included in the ITS itself, like for example in SYPROS.

6 List of tested Browsers

✓: tested ok. no: tested, but failed. –: browser/ OS combination not available for testing.

Webbrowser	Version	Win98/NT		Linux		Solaris/x86		Solaris/Sparc		Macintosh	
		Servl.	RMI	Servl.	RMI	Servl.	RMI	Servl.	RMI	Servl.	RMI
Netscape	4.04	✓	✓	–	–	✓	–	–	–	✓	no
Netscape	4.05	✓	✓	✓	✓	–	–	✓	✓	[illegible]	no
Netscape	4.7	✓	✓	✓	✓	✓	✓	–	–	[illegible]	no
Netscape	4.72	✓	✓	–	–	–	–	–	–	✓	no
I-Explorer	4.0	✓	no	–	–	–	–	–	–	–	–
I-Explorer	5.0[3]	✓	✓	–	–	–	–	–	–	–	–
Java Plugin	1.2	✓	✓	✓	✓	✓	✓	✓	✓	–	–

References

[1] Braun, A., "A Client-Server Model for an Intelligent Tutoring System" (transl.), Technische Universität München, Diploma Thesis, May 2000.

[2] Brügge, B., Dutoit, A.H., "Object-Oriented Software Engineering: Conquering Complex and Changing Systems", Prentice Hall, Upper Saddle River, NJ, Oct. 1999.

[3] Brusilovsky, P., Ritter, S., Schwarz, E., "Distributed intelligent tutoring on the Web", in: Proceedings of the 8th. World Conference of the AIED Society, Kobe, Japan, Aug. 1997.

[4] Dijkstra, E.-W., "Cooperating sequential processes", in: F. Genuys (editor), Programming Languages, Academic Press, New York, Oct. 1968.

[6] Duncan Davidson, J., "Java™Servlet Specification v.2.2", Sun Microsystems Inc., Oct. 1999.

[7] Eckert, C., "Matching Security Policies to Application Needs", Proceedings of the IFIP TC11 11th Internation Conference on Information Security, Cape-Town, South Africa, May 1995, p.237-254.

[8] Eckert, C., "On Security Models", Proceedings of the IFIP TC11 12th International Conference on Information Security, Samos, Greece, May 1996.

[9] Flanagan, D., "Java in a Nutshell", O'Reilly & Associates, third edition, 1999.

---

[3] RMI-Patch applied by Microsoft Service Pack.

BEST COPY AVAILABLE

[10] Gamma, E., Helm, R., Johnson, R., Vlissides, J., "Design Patterns -- Elements of Reusable Object Oriented Software", Addision Wesley Publishing Company, Reading Mass., tenth edition, 1997.

[11] Harrer, A., Herzog, Ch., "SYPROS Going IDLE -- from a Classical ITS to an Intelligent Distributed Learning Environment", in: Proceedings of ICCE '99, 7th International Conference on Computers in Education, p.836-839, Sep. 1999.

[12] Harrer, A., "Both sides of the coin -- blending cognitive and motivational aspects into a tutoring strategy", in: Proceedings of the International Conference on Computers in Education, ICCE97, p.188-195, 1997.

[13] Herzog, Ch., "Cooperation and competition of syntax-oriented and execution-oriented problem solvers in intelligent programming tutors", in: Artificial Intelligence in Education -- Knowledge and Media in Learning Systems, Proc. AI--ED 97, Kobe, Japan, p.309-316, 1997.

[14] —,"Java Native Interface", JDK Documentation, Sunsoft, 1996.

[15] Martha C. Polson, Richardson, J. J., "Foundations of Intelligent Tutoring Systems", Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1988.

[16] —,"Java Remote Method Invocation -- Distributed Computing for Java", Whitepaper, http://java.sun.com/marketing/collateral/javarmi.html, Sun Microsystems Inc., 1999.

[17] —,"Java Remote Method Invocation Specification", Revision 1.50, JDK 1.2, Sun Microsystems Inc., October 1998.

[18] —,"RMI-IIOP Programmer's Guide", http://java.sun.com/products/jdk/guide/rmi-iiop/rmi_iiop_pg.html, Sun Microsystems Inc. and IBM Corporation, 1999.

[19] Rumbaugh, J., Jacobson, I., Booch, G., "The Unified Modeling Language Reference Manual", Addision Wesley Longman, Inc., Reading Mass., Dec. 1998.

[20] Tanenbaum, A. S., "Modern Operating Systems", Prentice Hall Inc., 1992.

[21] McPherson, Scott, "Java Servlets and Serialization with RMI", in: Java Developer Connection, http://java.sun.com/developer/technical/Articles/RMI/rmi/index.html, Maczieg Software, 1999.

[22] —,"The Common Object Request Broker: Architecture and Specification", Object Management Group, Wiley, New York, 1995.

A Fuzzy-Based Assessment for Perl Tutoring System[1]

Tang Ya*, Keith C.C. Chan*, Albert Wu and Pinata Winoto*****
**Department of Computing*
Hong Kong Polytechnic University, Hong Kong
Email:{ csytang, cskcchan } @comp.polyu.edu.hk
***Faculty of Education*
Chinese University of Hong Kong, Hong Kong
Email: albertwu@cuhk.edu.hk
****School of Business and Management*
Hong Kong University of Science and Technology, Hong Kong
Email: pwinoto@ust.hk

In this paper, we present a fuzzy-based assessment for Perl Tutoring system. The Perl Tutor is implemented in a multi-domain framework so that it can teach target domain knowledge by giving supporting knowledge to reinforce the learning. In order to assess supporting knowledge, an assessment is performed *before* the tutoring begins. Its main purpose is to test student's previous declarative knowledge of computer programming. At the end of it, a directed tutoring graph will be generated to optimize the tutoring process.

Keywords: **fuzzy rule, assessment, student modeling, multi-domain tutoring**

1 Introduction

There exist many works on optimized assessment process concerned with the efficiency of testing and its completeness. *Granularity, prerequisite relationships, Bayesian propagation* and *neighborhood of knowledge states* are some of the successful attempts employed to increase the efficiency of testing [2,5,6,13,17]. Yet, even though they could increase the efficiency significantly, they still have too many burdens given the large knowledge spaces. Fortunately, not all the student models need to be precise to be useful [10]. To ease the burden to student modeling, a fuzzy approach has been used and has so far worked quite well [3,10,11].

The purpose of this paper is to present the fuzzy approach in the assessment of student's knowledge in the Perl Tutoring System [16], which teach programming language (Perl) by reinforcement from other supporting languages (C++ and/or Java). For the effectiveness of reinforcement, the system should quickly evaluate the student's knowledge of supporting languages. But the assessment needs not to be in high precision. Other works related to student modeling almost put their emphasis on the adaptive assessment *during* tutoring [14,15,17]. Yet due to the nature of our Perl tutor, we apply an assessment module *before* tutoring begins and it consists of two parts: questionnaire and testing. During the questionnaire part, students are asked to self-assess their knowledge by filling out a form provided by the system. In order to evaluate their statements, a testing part is given based on those statements. At the end of the assessment, the tutor will have a general picture of students' prior knowledge of supporting languages: with which part they are familiar etc. Since the goal of the assessment is only to get a rough knowledge states for supporting purpose, it should not take too long to complete. Thus, a coarse granularity with imprecise mastery level is appropriate.

In the next part of this paper, we briefly discuss the Perl tutoring system followed by the fuzzy logic. Then we will describe the questionnaire part and the testing part and end with discussion.

---

[1] The work related to this paper is funded under the Hong Kong Polytechnic University research grant No. PolyU5072/98E.

2 Overview of the Perl Tutoring System

Figure 1 illustrates the directed tutoring graph in the system [16]. The three pieces of knowledge items presented to students are: data type, logical operators and control structures. In the figure,

- Each vertex represents a sub-domain;
- Each pair of the sub-domain may be connected with a unidirectional or bi-directional arc.
- Each arc represents the relationship between two sub-domains.

Moreover, each sub-domain may consist of several vertices, which are the sub-sub-knowledge items of their parent domain. For example, under ' data type' , we also have ' integer' , ' float' , ' boolean' etc.

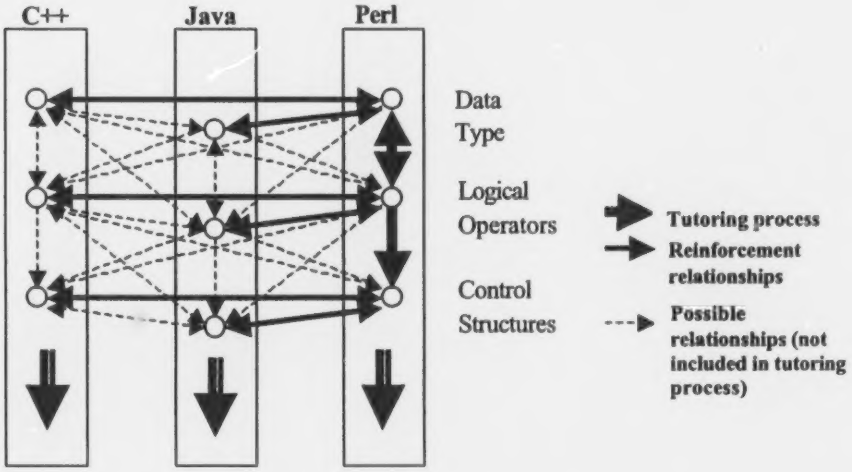

C++
Java
Perl
Data Type
Logical Operators
Control Structures
Tutoring process
Reinforcement relationships
Possible relationships (not included in tutoring process)

Figure 1 A Digraph of the Perl Tutoring System

C++ [1] and Java share many similarities with Perl, although they, of course, have their own features. See Table 1 for a comparison.

CDR terms (General)			Knowledge piece in PERL	Knowledge piece in C++	Knowledge piece in Java
Operators	Numeric operators		+,-,*,/, %, **	+,-,*,/, %,	+,-,*,/, %,
	Relational operators		<,<=,>,>=,<=>(**for numeric**) lt,le,gt,ge,cmp(**for string**)	<,<=,>,>=	<,<=,>,>=
	Equality operators		=,!=**(for numeric)** eq,ne**(for string)**	==,!=	==,!=
	Logical operators(binary)		&&,||	&&,||	&&,||
	Logical operators(unary)		!,	!,-	!,-
	Bit manipulation operators		&, |, ^, ~	&, |, ^, ~	&, |, ^, ~
	Bit shift operators		<<, >>	<<, >>	<<, >>, >>>
	Auto-increment & auto-decrement operators		++,--	++,--	++,--
	Speical operators	Conditional operators	?:	?:	?:
		Other operators	•,x(string operators) , **>,\(dereference/reference operator**	Sizeof ,	Instanceof
	Multiple operators			+=, -=, *=, /=, %=, &=, |=, <<=, >>=, ++=, --=	+=, -=, *=, /=, %=, &=, |=, <<=, >>=, >>>=, ++=, --=
Control structures			If,if/else, unless/else, While, do/while, for, continue, goto Notes:labled loops can be used within for, while, or do	If, if/else, while, do/while, for, continue, goto, switch, break	if, if/else, while, do/while, for, continue, break, switch Notes:labled loops can be used within for, while, or do
	Special structures		**Foreach** **Redo**	Exit	

Table 1 Similarities and differences in C++, Java and Perl

CDR represents ' cross-domain reference' which serves as a dictionary for the domains. It is composed of basic terms used across the computer language regardless of which language is being referred. If the student has learned computer language before, he will develop a clear picture of the terms or concepts used, which serves as a guide for the learning of Perl. Besides, he will also integrate his former learning into his current. Through this knowledge transfers, the time spent on learning Perl will be greatly reduced [8].

Before tutoring begins, a weight is assigned to every direction of arc that represents the easiness of the acquisition of one sub-domain (target) after acquiring another (source). Since different students have different knowledge levels, the weight assigned to the same arc may not be the same. Thus, the weight across domain is jointly determined by the student model and the characteristics of knowledge (for detailed explanation, refer to [16]), i.e.,

$$w_{ij} = f(\mathbf{d}_{ij}, \mathbf{m}_{ij})$$

Where, w_{ij} is the weight of arc from i to j.

f: $R^n \times R^m \rightarrow R_+$ is a non-decreasing function.

$\mathbf{d}_{ij}$ is an n-dimensional vector representing the similarity of i and j. $\mathbf{m}_{ij}$ is an m-dimensional vector representing the student model, i.e., the student' s knowledge level of i.
The dimension of $\mathbf{d}_{ij}$ and $\mathbf{m}_{ij}$ depends on the number of attributes considered. Moreover, the value of $\mathbf{d}_{ij}$ is predetermined and the value of $\mathbf{m}_{ij}$ is determined based on the student model. Thus, the system would carry an assessment module to test the knowledge of a student towards a specific supporting domain knowledge *before* tutoring begins. In this paper, we focus on the determination of $\mathbf{m}_{ij}$.

3 The Assessment Model–A Fuzzy Approach

Since the main purpose of the model is to test student' s overall abilities, it is not necessary for us to gain a very accurate picture of it (although it helps). And somehow we also cannot gain a clear picture of student history. Thus, we choose a fuzzy approach in analyzing the student' s performance, and we believe that the imprecise assessment of the student' s prior knowledge level is adequate.

3.1 The ' neighborhood of knowledge states'

The *knowledge state* has been defined as the subset of knowledge items from a large item pool that can be mastered by students [4]. Remember that knowledge items in different domains are identified by their names, which in turn are determined by a cross-domain vocabulary. Besides, each item is characterized by its relationship with other items. The neighborhood of a knowledge state was defined by Falmagne and Doignon [7] as all other states within a distance of at most one. It has been utilized for adaptive assessment by Dowling *et. al.* [6]. In our system, we will not measure the exact distance within knowledge items, but we adopt it from another perspective. We define the neighbors of a knowledge item as the possible knowledge items which could be mastered in association with it. Let us have a look at an example.

Example 1.

1. ' < ' ,' <=' represent ' less than' and ' less than or equal to' respectively, and they are relational operators.
2. ' > ' ,' >=' represent ' greater than' and ' greater than or equal to' respectively.
3. ' == ' ,' ! =' represent ' equal to' and ' not equal to' respectively, and they are equality operators.
4. ' < ' ,' <=' can be used for both numeric and strings.
5. ' > ' ,' >=' can be used for both numeric and strings.
6. ' == ' ,' ! =' can be used for both numeric and strings.
7. Numeric is data type.
8. Strings are data type.
9. The relational and equality operators can be used for all data types, numbers, expressions or their combinations.

Let $M_S(X)$ denotes the student is sure to have mastered X. And $M_l(Y)$ denotes the student is likely to have mastered Y. Where X,Y are sets of knowledge items. Then,

$M_S(X)$□ $M_l(Y)$ can be interpreted as "if the student is sure to have mastered X, then he/she is likely to have mastered Y."

Then we will have:
1. $M_S(1)$□ $M_l\{2,4,7,8\}$
2. $M_S\{3\}$□ $M_l\{6,7,8\}$
3. $M_S\{4,5,6\}$□ $M_S\{7,8\}$
4. $M_S\{9\}$□ $M_S\{1,2,3,4,5,6,7,8\}$

For example, if the student knows well how to make comparisons for numeric and strings, then we assume that he/she is sure to have mastered: what is numeric, what is a string and the usage of the operators. Although we cannot determine that whether he masters other data types or not (that is, he is likely to have mastered other data types such as float etc), we can assess student' s knowledge state without having to extensively test his abilities of each knowledge item he/she may have learned. Therefore, test items in our model may test knowledge items in a wider ranger than similar work by Collins *et. al.* [2].

3.2 Fuzzy Logic

To express precisely the notion "sure", "likely" or "unlikely", we adopt fuzzy set methods and therefore using fuzzy rule for the inferences. For example, we define

Answer = {True, False}. And A1, A2 $\subset$ Answer, thus

$$Ai = \mu_{Ai}(T)/True + \mu_{Ai}(F)/False$$

Confidence = {unlikely, likely, sure}. And B1, B2 $\subset$ Confidence, thus

$$Bi = \mu_{Bi}(u)/unlikely + \mu_{Bi}(l)/likely + \mu_{Bi}(s)/sure$$

Assume we have two rules: R1: A1 → B1 and R2: A2 → B2

Then, by Mamdani' s direct methods:

$$B' = A' \circ R$$

Where, $R = R1 \cup R2$

$$Ri = \left(\begin{array}{ccc} \mu_{Ri}(T,u) & \mu_{Ri}(T,l) & \mu_{Ri}(T,s) \\ \mu_{Ri}(F,u) & \mu_{Ri}(F,l) & \mu_{Ri}(F,s) \end{array}\right] \quad \text{and} \quad \mu_{Ri}(x,y) = \mu_{Ai}(x) \wedge \mu_{Bi}(y)$$

Note here that all operators used, such as: +, /, $\subset$, $\wedge$, $\cup$, and o, are defined in fuzzy domain.[2]

To illustrate it, let us assume that A1 is "doing well in bit shift operator", A2 is "doing bad in bit shift operator", B1 is "understand bit manipulation if doing well in bit shift operator", and B2 is "understand bit manipulation if doing bad in bit shift operator". Then, we can assign values such as:

A1 = 1.0/T A2 = 1.0/F

B1 = 0.5/l + 0.5/s B2 = 1.0/u + 0.1/l

And satisfied: R1: A1 → B1 and R2: A2 → B2. Thus,

$$R1 = \begin{array}{cc} & \begin{array}{ccc} \mu_{B1}(u) & \mu_{B1}(l) & \mu_{B1}(s) \\ 0 & 0.5 & 0.5 \end{array} \\ \begin{array}{cc} \mu_{A1}(T) & 1.0 \\ \mu_{A1}(F) & 0 \end{array} & \left[\begin{array}{ccc} 0 & 0.5 & 0.5 \\ 0 & 0 & 0 \end{array}\right] \end{array}$$

$$R2 = \begin{array}{cc} & \begin{array}{ccc} \mu_{B2}(u) & \mu_{B2}(l) & \mu_{B2}(s) \\ 1.0 & 0.1 & 0 \end{array} \\ \begin{array}{cc} \mu_{A2}(T) & 0 \\ \mu_{A2}(F) & 1.0 \end{array} & \left[\begin{array}{ccc} 0 & 0 & 0 \\ 1.0 & 0.1 & 0 \end{array}\right] \end{array}$$

---

[2] Many books [18,19,20] in fuzzy set theory provide good explanations on these operators. We are not going to explain it further in this paper due to limited space.

With two rules, the fuzzy relation Ri is made from the implication Ai → Bi (in this case, i=1,2). The compiled fuzzy relation R is given as Mamdani' s method:

$$R = R1 \cup R2, \text{ computed as:}$$

$$R = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 1.0 & 0.1 & 0 \end{bmatrix}$$

Now, assume after a series of testing, a student performance show A' = 0.9/T + 0.2/F in doing bit shift operator. Then, we can calculate his performance in bit manipulation as:

$$B' = A' \mathrm{o} R$$

$$= \begin{array}{cc} T & F \\ [0.9 & 0.2] \end{array} \quad \mathrm{o} \quad \begin{array}{cc} & \begin{array}{ccc} u & l & s \end{array} \\ \begin{array}{c} u \\ l \end{array} & \begin{bmatrix} 0 & 0.5 & 0.5 \\ 1.0 & 0.1 & 0 \end{bmatrix} \end{array}$$

$$\begin{aligned} = \; & [(0.9 \wedge 0) \vee (0.2 \wedge 1.0), \\ & (0.9 \wedge 0.5) \vee (0.2 \wedge 0.1), \\ & (0.9 \wedge 0.5) \vee (0.2 \wedge 0)] \end{aligned}$$

$$= [0.2 \quad 0.5 \quad 0.5]$$

$$B' = A' \mathrm{o} R = 0.2/u + 0.5/l + 0.5/s$$

Which shows 0.5 likely to understand, 0.5 surely to understand and only 0.2 unlikely to understand bit manipulation.

4 Questionnaire and Testing

The questionnaire part consists of a series of knowledge items to be checked by students. The knowledge items are grouped into several groups based on the their similarities and difficulties. Then, students are asked to fill the form about their mastery level in each group. Five grades are provided for each answer, i.e., very familiar, familiar, moderately familiar, not familiar, and never heard. After students provided their answers, the system retrieves a series of testing questions based on the difficulty (upper limit) of students' answers, especially for the items marked 'moderately familiar'. But it does not mean that the presumably mastered items are not tested at all. Even the items marked 'very familiar' will be tested, but with a very low probability. Testing could be in the forms of short program lists or short questions, which are made as short, clear, and simple as possible. The reason is to avoid noise or errors which do not come from student knowledge iteself. In order to avoid ambiguity in judging knowledge level when the question is not answered well, every question only consists few higher level concepts to be handled.

Moreover, an average of membership value is used if the same item occurs in several questions. (We can use Bayesian update but with higher cost, i.e., to set all the conditional probability among every question).
For example, if from question 1, 2 and 3, a student performance on 'bit manipulation' shows

0.8/T + 0.2/F, 0.9/T + 0.3/F, and 1.0/T + 0.1/F respectively,

then the overall performance is, simply, the average, i.e., 0.9/T + 0.2/F.

If the question needed does not exist in the database, then a similar question is retrieved. The measure of similarity is based on the maximum number of high level concept appeared.

Prerequisite relationship

In addition to the neighborhood relationships, prerequisite relationships are also applied. The prerequisite relationship provides not only test item ordering criteria in a "strong" sense, but also in a "weak" sense. In ordinary prerequisite criteria P(A, B) denotes "A is prerequisite of B". In our extended criteria, we introduce A' as:

If A' is closely related to A and $M_S(A')$ □ $M_S(A)$
then we have $P'(A', B)$, that is, A' is weakly prerequisite of B.

So, if students have mastered item A', we have: they are sure to have mastered B without testing whether they have mastered item A or not. By doing this, we can largely tighten the testing items and thus save more time.

5 Discussion

To know student's learning history and his knowledge level, we cannot ask them too detailed questions in order to gain a more full picture of their knowledge state (although it helps) since it will make student modeling itself a kind of a complex system. But we need them to aid in the assessment, so how much trust should we have in the student's own assessment? This is the question we need answer before we proceed. In our system, we will not generate the tutoring graph solely based on their answers. Our solution is to test by giving them several pre-stored test items: if they can write out the outcome correctly, we assume that he has mastered the knowledge pieces and rules needed for this program.

Thus, the assessment will proceed. Test items need not to be like traditional testing questions in classrooms. They can be mini-programs or short questions provided that they can be used as a guide to assess students' mastery level of declarative knowledge.

Furthermore, we also should consider the nature of the language. For example, If the student has studied both Prolog and Java before, considering the respective relationship of them with Perl, we will still use Java as supporting knowledge because it is closer to Perl. This factor is called Knowledge Relation (K-R), and it will be assigned to $\mathbf{d}_j$.

At the end of the self-assessment section, a directed tutoring graph is generated. And student will be tutored based on it.

Currently, we are constructing the fuzzy rules which are applied for the assessment module, followed by the implementation and evaluation of it.

References

1. Capper, D.M. "Introducing C++ for Scientists, Engineers and Mathematicians", U.K., Springer-Verlag, (1994).
2. Collins, J.A., Greer, J.E. & Huang, S.X. "Adaptive assessment using granularity hierarchies and Bayesian nets", Proceedings of the 3rd International Conference on Intelligent Tutoring Systems (ITS'96), Montreal, Canada, LNCS 1086, (1996).
3. Derry, S.J. & Hawkes, L.W. "Toward fuzzy diagnostic assessment of metacognitive knowledge and growth", Proceedings of the Annual Meeting of the American Educational Research Association, San Francisco, CA, (1992).
4. Doignon, J.-P. & Falmagne, J.-C. "Spaces for the assessment of knowledge", International Journal of Man-Machine Studies, Vol 23, pp.175-196, (1985).
5. Dowling, C.E. & Kaluscha, R. "Prerequisite relationships for the adaptive assessment of knowledge", Artificial Intelligence in Education, pp. 43-50, AACE: Charlottesville, VA, (1995)
6. Dowling, C.E., Hockemeyer,C., Ludwig, A.H. "Adpative assessment and training using the neighbourhood of knowledge state", Proceedings of the 3rd International Conference on Intelligent Tutoring Systems (ITS'96), Montreal, Canada, LNCS 1086, (1996).
7. Falmagne,J.-C. & Doignon,J.-P. "A Markovian procedure for assessing the state of a system", Journal of Mathematical Psychology, Vol. 32, pp. 232-258, (1988).
8. Harvey, L. & Anderson, J.R.. "Transfer of declarative knowledge in complex information processing domains", http://act.psy.cmu.edu/ACT/papers/Harvey_Anderson95.html, (accessed on August 1, 2000).
9. Juliano,B.A. & Bandler,W. "Tracing Chains-of-Thought", Heidelbert: Physica-Verlag, (1996).
10. Katz, S., Lesgold, A., Eggan, G., & Gordin, M. "Approaches to student modeling in the Sherlock tutors", Proceedings of the Third International Workshop on User Modeling, Germany, pp. 205-230, (1992).

11. Lesgold, M., Eggan, G., Katz, S. & Rao, G. "Possibilities for assessment using computer-based apprenticeship environments", Cognitive Approaches to Automated, Hillsdale, Lawrence Erlbaum Associates, (1991).
12. Matsubara, Y., Nagamachi, M. "Motivation system and human model for intelligent tutoring", Proceedings of the 3rd International Conference on Intelligent Tutoring Systems (ITS'96), Montreal, Canada. LNCS 1086, (1996).
13. McCalla, G. I., Greer, J. E. "Granularity-based reasoning and belief revision in student models", Student Modelling: The Key to Individualized Knowledge-Based Instruction, pp. 39-62. Springer-Verlag, Berlin, (1994).
14. Nguyen-Xuan, A., Joly, F., Nicaud, J.F., Gelis, J.M. "Automatic diagnosis of the student's knowledge state in the learning of algebraic problem solving", Proceedings of the Conference on Artificial Intelligence in Education, Japan, (1993).
15. Sime, J.-A. "Modeling a learner's multiple models with Bayesian belief networks", Proceedings of the Conference on Artificial Intelligence in Education, Edinburgh, Scotland, pp. 426-432, (1993).
16. Tang, Tiffany Ya. & Wu, Albert. "The implementation of a multi-agent intelligent tutoring system for the learning of computer programming", To appear in the 16th IFIP World Computer Congress—International Conference on Educational Uses of Communication and Information Technologies (ICEUT 2000) Beijing, China, (2000).
17. Villano, M. "Probabilistic student models: Bayesian belief networks and knowledge space theory", Proceedings of the 2nd International Conference on Intelligent Tutoring Systems (ITS'92), Montreal, Canada, pp. 491-498, (1992).
18. Zadeh, L.A "Fuzzy sets", Information and Control, Vol. 8, pp. 338-353, (1972).
19. Zadeh, L.A. "A fuzzy-set-theoretic interpretation of linguistic hedges", Journal of Cybernetics, Vol. 2, pp. 4-34, (1973).
20. Zhu, Q. & Lee, E.S. "Fuzzy and Evidence Reasoning", Physica-Verlag, Germany, (1995).

A Learning Environment for Problem Posing in Simple Arithmetical Word Problems

Akira Nakano, Naoki Murakami, Tsukasa Hirashima and Akira Takeuchi
Kyushu Institute of Technology Department of Artificial Intelligence
680-4 Kawazu, Iizuka 820-8502, Japan
E-mail : nakano@minnie.ai.kyutech.ac.jp

Several researchers indicate that to pose arithmetical word problems is an important way to learn arithmetic. However, the problem posing practice actually is not popular. In this paper, we describe an Intelligent Learning Environment which realizes the problem posing practice. In the problem posing practice, the learners pose problems by using the tools provided by the ILE. The ILE has a facility to diagnose the problems posed by the learners. By using the result of the diagnosis, the ILE indicates whether the problems are correct or not, helps the learner to correct the wrong problems, and provides the next step of problem posing. We used the ILE in three different situations for evaluation. The subjects were elementary school teachers and elementary school students. We also report the results of the evaluation. In the ILE, the interface was implemented in Java, and the diagnosis module was implemented in Prolog. So it can be used on World Wide Web. The current environment deals with simple arithmetical word problems.

Keywords: **intelligent learning environment, problem posing, intelligent tutoring system, interactive education, World Wide Web**

1 Introduction

The main purpose of the practice to solve arithmetical word problems is to make learners recognize the relations between concepts and numerical relations, and master to use the relations. Although the problem solving practice is the most popular way, it is not the only way. Several researchers indicate that to pose arithmetical word problems is also effective. However, the problem posing practice actually is not popular.

The main reason is that the problem posing practice is strongly required teachers to deal with each learner individually in comparison with the practice of problem solving. We aim to realize computer-based learning environments for the problem posing practice [1]. For the problem solving practice, many ILEs are developed so far [2-6]. However, there are few ILEs for the problem posing practice until now.

This paper describes an Intelligent Learning Environment for the problem posing practice for simple arithmetical word problems that can be solved by the addition of one time or the subtraction of one time. The main characteristic of the ILE is the function to diagnose the posed problems. By using the results, the ILE indicates errors in the posed problems and suggests that the next step of problem posing.

Interface of the ILE was implemented in Java, and the diagnosis module was implemented in Prolog. Therefore, if only users have a computer connected to Internet with a popular internet browser, they can use the ILE through WWW: E-mail: nakano@minnie.ai.kyutech.ac.jp http://www.minnie.ai.kyutech.ac.jp/~nakano/problem-posing.shtml (currently Japanese only).

In this paper, the first, the necessity of problem posing and an Intelligent Learning Environment for it are described. Then, interface and diagnosis module of the ILE are explained. The results of preliminary evaluation of the ILE are also reported.

2 Background

2.1 The necessity of an ILE for problem posing

Several researches about problem posing of arithmetical word problems suggested that problem posing was important to learn arithmetic, for example, analysis and investigation about the task of problem posing [7,8], investigation about effect of the problem posing practice [9], investigation in the problem posing practice at arithmetic class [10,11]. Besides, the Curriculum and Evaluation Standards for School Mathematics (in USA, 1989), and Professional Standards for Teaching Mathematics (in USA, 1991) also indicated that it was important for learners to experience to pose problems.

However, the practice actually is not popular in arithmetic class in comparison with problem solving practice. In the practice of problem solving, every problem has an answer and one or a few solution methods. Therefore, the teachers can easily judge the results of problem solving by learners. Then when the answer is wrong, to tell the correct answer or the solution method is not meaningless.

In contrast with problem solving practice, to prepare every correct problem in the problem posing practice is very difficult. Besides, the correct problem that a learner is trying to pose, after depends on the wrong problem posed by the learner. Therefore, the teachers have to examine each problem whether the problem is correct or not, and where of the problem is wrong.

Based on this consideration, we believe that to realize an ILE for problem posing with problem diagnosis function is the promising way to make learning by problem posing popular.

2.2 The problem posing dealt in the ILE

Silver has noted that the term "problem posing" is generally applied to three quite distinct forms of mathematical cognitive activity [12]. They classified three types of problem posing: (1) presolution posing, in which one generates original problems from a presented stimulus situation, (2) within-solution posing, in which one reformulates a problem as it is being solved, and (3) postsolution posing, in which one modifies the goals or conditions of an already solved problem to generate new problems. The problem posing deal in our ILE is (2) within-solution posing. In the ILE, because, in the ILE, first, a learner decides a calculation formula to solve the problem, and next, he/she is trying to pose problem solved by the calculation.

Currently, the LIE can deal with only Change-Problem[13]. In Change-Problem, the quantity in the initial situation is changed to the quantity in the final situation by the change action. The Change-Problem usually consists of three sentences: the first sentence describes the initial situation, the second sentence describes the change action, and the third sentence describes the final situation. Therefore, we prepare a "problem template" that composed of three single sentence templates. By filling in the blanks of three single sentence templates, the problem is completed.

In the ILE, the template of Chang-Problem is composed of the tree single sentence templates that describe: initial situation, change action, and final situation, respectively. The initial situation has the four information: "owner", "object", "number", and "unit". This means that "owner" has "object" and the number of "object" is "number", then, the unit of the number is "unit". The change action has the five information: "actor", "object", "number", "unit", "action". Several actions, for example, "take" has two more information: "from" and "to". The final situation has the four information: "owner", "object", "number", and "unit".

3 ILE for problem posing

3.1 Configuration of the ILE

The current version of the ILE consists of clients and server shown in Figure 1. A client is an interface of the ILE. The interface provides learners the tools to pose problems and gives them guidance to promote problem posing. Inter face is explained in more details in this section 3.2.

The server has two modules: the one is Problem Diagnosis Module and the other is Advice Generator. First, the ILE receives the posed problem, and diagnoses it in Problem Diagnosis Module. Next, in Advice Generator, the ILE generates advice for each learner by using the result of diagnosis. These are explained in more details in this section 3.3.

Because the ILE deals with several learners by one server, the ILE manages ID, PW, and Learner Model in Private Information Manager.

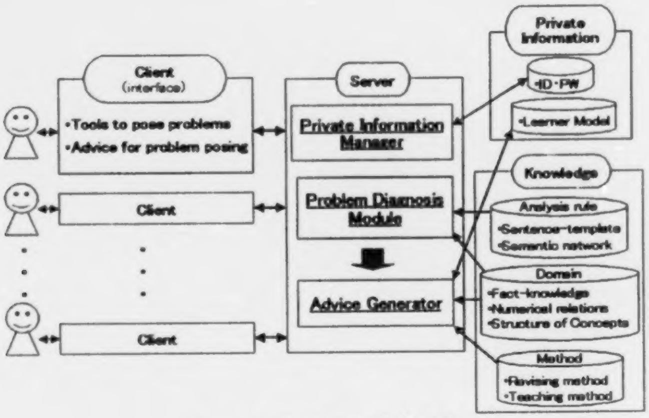

Client (interface)
•Tools to pose problems
•Advice for problem posing
Client
Client
Server
Private Information Manager
Problem Diagnosis Module
Advice Generator
Private Information
•ID·PW
•Learner Model
Knowledge
Analysis rule
•Sentence-template
•Semantic network
Domain
•Fact-knowledge
•Numerical relations
•Structure of Concepts
Method
•Revising method
•Teaching method

Figure 1: The frame of the ILE

3.2 Interface

Figure 2 shows the interface of the ILE. Current interface deals with only Japanese. In Figure 2, Japanese was translated into English for this paper. The parts of the interface are expressed as follows.

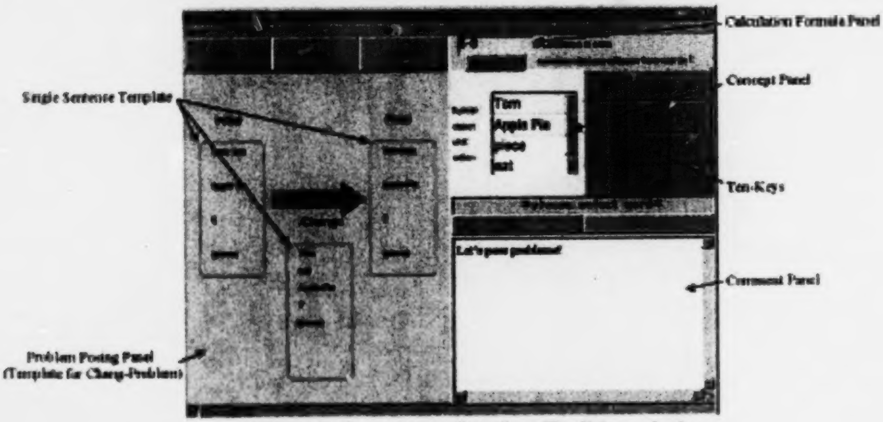

Calculation Formula Panel
Concept Panel
Ten-Keys
Comment Panel
Single Sentence Template
Problem Posing Panel
(Template for Change-Problem)
Tom
Apple Pie
piece
eat
Let's pose problems!

Figure 2: the prototype Interface (English version)

- Calculation Formula Panel

In this panel, a learner gives calculation formula. The learner poses problems which can be solved by this calculation formula.

- Concept Panel

This panel provides concepts to fill in blanks of sentences (three single sentences templates). The concepts that are provided in the Concept Panel are classified in the five categories: "human", "object", "unit", "action", "number".

- Ten-key

BEST COPY AVAILABLE

Numerical values are put into blanks of sentences with Ten-key.

- Problem Posing Panel

In the current version, this panel provides the template of Change-Problem. The ILE asks a learner to fill in blanks of sentences. In the order of the blanks, the ILE gives questions. By answering the questions, the blanks are filled in. Here, the learner has to select concepts from Concept Panel. By using Figure 3, posing a sentence of initial situation in Chang-Problem is explained. The left side of the figure shows questions.

For example, the initial situation in Chang-Problem is composed of four elements: "owner", "object", "number" and "unit". So, the ILE asks the learner "Who has?", "What the person has?", "How many?", "What is unit?". The learner also should decide what number is the answer by selecting the question mark in Ten-key.

The right side of the figure shows an example which the learner answered the questions. The result shows "Tom has 5 pieces of Apple Pies".

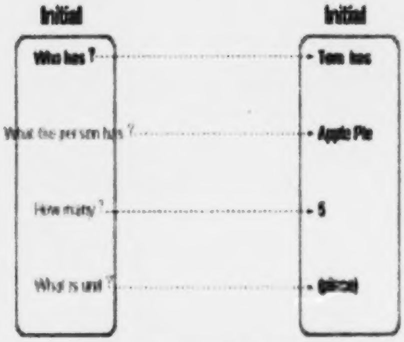


Figure 3: An example of posing sentence by using Single Sentence Template

By answering the all questions, learners pose problems For example, Figure 2 shows the correct problem in Problem Posing Panel: the initial situation is "Tom has 5 pieces of Apple Pies", the change action is "Tom eats the 3 pieces of Apple Pies", and the final situation is "How many pieces of Apple Pies does Tom have?".

- Comment Panel

This panel shows advice and suggestion massages that are generated based on the diagnosis of the posed problems.

3.3 Problem posing in the ILE

A learner poses a problem by the following process.

(1). Giving a calculation formula

First, the learner gives a calculation formula. The calculation formula consists in three elements. That is, two operands and an operator. Because the calculation formula is the way to get the answer of the problem, we call it solution.

The solution can be applied to several numerical relations. For examples, if the learner assigned "5-3" to the solution, the solution can be applied to the following four numerical relations: (a) "5-3=X", (b) "5-X=3", (c) "3+X=5", (d) "X+3=5" (the current version of the ILE only handles natural numbers). Here, numerical relation (a) means the answer is the number in the final situation, numerical relation (b) and (c) mean the answer is the number in the change action, and numerical relation (d) means the answer is the number in the Initial situation.

(2). Selecting concepts from Concept Panel and combining them with the template of Change-Problem

The template has several blanks, and the ILE asks the learner to pose a problem by filling the blanks with the concepts. Then, if the learner selected a concept from the set of wrong concepts, the ILE can give the learner feedback, which suggested that the concept is wrong.

(3). Request to diagnose a problem

When the learner clicks the "diagnosis button", the problem is sent to the server and is diagnosed.

(4). Revising the wrong problem by using the suggestion given in the Comment Panel

When the posed problem is wrong, the learner receives feedback that indicates an error at Comment Panel.

The ILE generates the message by using the result of the diagnosis.

(5). Posing the new problem by using the suggestion

When the learner posed the correct problem, the learner receives feedback which is suggests to pose the new type of problems.

3.4 Problem Diagnosis Module and Advice Generator

Problem Diagnosis Module and Advice Generator are functions of the server in the ILE. Problem Diagnosis Module diagnoses problems sent by the client, and Advice Generator generates messages that are provided for each learner.

The ILE, first, diagnoses a single sentence and then diagnoses the problem composed of three sentences, and compares the solution given by a learner with the problem posed by the learner. In the first step, the module has knowledge about acceptable sentences (initial situation, change action, final situation). We call each sentence "basic relation", and the knowledge "single sentence schema". The single sentence schema checks each basic relation to find the errors in a sentence.

In the second step, the relation among the sentences is diagnosed. The module has the knowledge about acceptable relations among basic relations. We call the knowledge "problem schema". The problem schema checks the numerical relation between the sentences to find the wrong sentence in the problem.

In the third step, the relation between the solution and the problem is diagnosed.

In the following section, the diagnosis process is explained. Then, the feedback made by the diagnosis result is presented.

3.4.1 Diagnosis of the posed problems

Diagnosis of the posed problems is carried out in three steps: the first step is the diagnosis of a single sentence. The second step is the diagnosis of the problem composed of three sentences. The third step is the diagnosis of the relation between the problem and the solution.

(1). Diagnosis of a single sentence

In this diagnosis, two types of errors are detected: (1-a) errors in the relation between object and action, and (1-b) errors in the relation between object and number. Here, Mismatch of blanks (that is, object blank or action blank and so on) and concepts is already checked in the interface.

An example of (1-a) is a sentence that "Tom eats his 2 sheets of postcards." "Tom has 5 cups of apple pies" is an example of (1-b). These errors are detected by checking with sentence schema in that the acceptable relations between object and action or object and number are described.

(2). Diagnosis of problem

In this diagnosis, three types of errors are detected: (2-a) errors in the final situation, (2-b) errors in the change action and (2-c) no relation errors. (2-a) means that the initial situation can be changed by the change action, but cannot be changed to the final situation. (2-b) means that the initial situation can be changed to the final situation, but cannot be changed by the change action. (2-c) means that the initial situation cannot be changed by the change action and to the final situation. These errors are detected by comparing by problem schema in that the acceptable relations among the situations and the change action are described.

An example of (2-a) is the problem composed of the following three sentences: "Tom has 5 pieces of apple pies", "Nancy eats Tom's 3 pieces of apple pies" and "how many pieces of lemon pies does Tom have?" An example of (2-b) is the problem composed of the following three sentences: "Tom has 5 pieces of apple pies", "Nancy eats her 3 pieces of apple pies, and "how many pieces of apple pies does Tom have?"

(3). Diagnosis of the relation between the problem and the solution

The diagnosis module can generate an equation from the problem. In this diagnosis, first, the module solves the equation. Then the calculation to derive the answer is compared with the calculation posed by the learner as the solution. When the two calculations do not correspond, an error in the relation between the problem and the solution is detected.

3.4.2 Feedback for the client

(1). Indication of an error

If the diagnosis module finds an error, the ILE indicates it. Even if the problem includes several errors, the

ILE indicates the error detected first.

(2). Suggestion of the next step of problem posing

The ILE suggests the next step of problem posing when the posed problem is the correct one. In the diagnosis, the module diagnoses not only whether the problem is correct or not, but also what concepts, actions or equations are used in the problem. Based on the results, the ILE can suggest more difficult problem posing by specifying concepts or an equation type to be allowed to use in problem posing.

4 Preliminary evaluations

A prototype of the ILE has been already developed. We used it in three different situation for evaluation, as follows: (1) Use by teachers of the elementary school, (2) Use by students of elementary school in arithmetic classes, (3) Use by students of elementary school outside the class.

In (1), we asked the teachers to evaluate the ILE from the viewpoint of teaching. Then, two of them permitted us to use the ILE in their arithmetic class. So, we had two opportunities to evaluate the ILE in the second situation. In (2), we asked the students of elementary school to pose arithmetical word problems with the ILE in two arithmetic classes. In the trial, although we collected the answers for our questionnaires, we failed to record logs of problem posing. Therefore, we could not get the data about the number of posed problems, and the students behave for feedback from the ILE. In (3), we gathered several students again, and asked them to use the ILE out of class. Here, the students used the ILE for the first time.

In this section, we report these results.

4.1 Use by the teachers of the elementary school

To evaluate a learning environment, the evaluation by teachers is important. We asked five teachers of elementary school to use the ILE. After they posed several problems by using this ILE, we asked them several questions. The questions are as follows: (1) How do you evaluate the effect of problem posing to learn arithmetic? (2) How do you evaluate the way of problem posing used in the ILE? (3) How do you evaluate the interface? (4) How do you evaluate the indications for the errors in posed problems? (5) How do you evaluate the advises to suggest the next step of problem posing? Table 1 shows the results.

Table 1-(1) means that all teachers think to learn arithmetic by using problem posing is effective. Table 1-(2) suggests that the ILE realizes an useful environment for learning by problem posing. Two teachers out of three teachers who answered "Good" to the question (2), gave us opportunities to use the ILE in classes. A few teachers also indicated that the limitation of concepts that were allowed to use in problem posing should be revised. This is one of our future works. In Table 1-(3), three teachers answered "So-so". The result means that the interface is not always easy to use. In Table 1-(4), four teachers answered "Good". The result suggests that the indications for the errors in the posed problem are acceptable. However, several teachers also indicated that the sentences of the indications may be difficult for elementary students. In Table 1-(5), the all teachers answered "Good". This result means that the teachers think the suggestions to make learners progress the next step of problem posing adequately support learning by problem posing.

Table 1: Evaluation of the teachers

	Good	So-so	Bad
(1)	5	0	0
(2)	3	2	0
(3)	2	3	0
(4)	4	1	0
(5)	5	0	0

4.2 Use by the students of elementary school in arithmetic classes

We used the ILE in two classes: the one was composed of 25 students in third grade and the other was composed of 30 students in fifth grade. In each class, 15 minutes were used to explain the use of the ILE, and 20 minutes were used for the problem posing practice with the ILE. In this problem posing practice, students were two people one set, and they operated one personal computer with two. Then two assistants assisted them to operate the ILE in the experiments.

Table 2: Evaluation of the

		Yes	No Answer	No
Third -grad	(1)	22	0	2
	(2)	22	0	2
Fifth-grad	(1)	27	2	1
	(2)	25	4	1

We asked two questions after the problem posing practice: (1) Are you interested in problem posing by using this ILE? (2) Do you want to pose more problems by using this ILE? The result is shown in Table 2.

The results suggested that most students were interested in problem posing with the ILE. But we were not able to get enough data to confirm that the students pose problem well.

4.3 Use by the students of elementary school outside the class

Subjects were one student of fourth-grade, and threes students of sixth-grade in elementary school. In the experiment, we used 15 minutes in the demonstration of this ILE, and 25 minutes in the problem posing practice. The results were as follows. In Table 3, *Diagnosis* indicates the number of time of request to diagnose.

Table 3: Logs of the problem posing

Diagnosis	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
i	α-C	β-C	γ-W	γ-W	γ-W	γ-W	γ-W	γ-W	γ-W	γ-C
ii	α-C	α-C	β-W	β-C	γ-W	γ-W	γ-W	γ-W		
iii	α-C	α-C	β-C							
iv	α-C	β-C	γ-W	γ-W	γ-W	γ-W	γ-W	γ-W	γ-C	

α: $A \pm B = X$, β: $A \pm X = C$, γ: $X \pm B = C$ (A,B,C are numerical values. X is a variable)

C: Correct, W: Wrong

In Table 3, equations named by Greek (α, β, γ) specify the type of problem posed by subject. "A" is the number in the initial situation, "B" is the number of the change action, and "C" is the number in the final situation. "X" is the number that is derived by the solution. In the α type, the answer is in the final situation. So this type of problem is the easiest one. In the β type, the answer is in the change action. In the γ type, the answer is in the initial situation. In this order, problems become difficult. The ILE can judge not only "C (correct)" or "W (wrong)", but also the type of problem whenever the student requests the diagnosis.

In Table 3, three subjects (i, ii, iv) tried to pose the problems of the all types, and subject-iii tried to pose the two types of the problems. The subject-i posed the wrong problem of the γ type on the 3rd request to diagnose in the practice. And the subject was repeating to revise it in seven times. As a result, the subject posed the correct problem of the γ type in the 10th trial. And the subject-ii posed the wrong problem of the β type on the 3rd request to diagnose in the practice, then the subject posed the correct problem of the β type in the 4th trial. And the subject-iv posed the wrong problem of the γ type on the 3rd request to diagnose, then the subject posed the correct problem of the γ type in the 9th trial, too. But, the subject-ii gave up to correct the wrong problem of the γ type, although s/he was repeating to revise the wrong problem in three times. The results suggest that the feedback is effective to forward the learner to revise the wrong problem.

In the current ILE, if a learner corrected the problem, the ILE suggests the next step of problem posing. The first step is problem of the α type. the second step is problem of the β type. and the third step is problem of the γ type. In Table 3, all subjects follow the suggestion. In the results, when a learner posed a correct problem, the learner can not poses only the same type of problem again, but also other types of problem by using the feedback. This suggests that the feedback is also effective to advance the next step of problem posing.
Conclusions

5 Conclusion

In this paper, we described ILE for problem posing in simple arithmetical word problems. The ILE provides the template to pose Change-Problem in current version. And the ILE can diagnose the problem that learners fill blanks of the template with several concepts, values, and question mark. Besides, the ILE can support each learners by using the results of diagnosis. We used the ILE in three different situations for evaluation. In the results, we consider that this research provides basis functions to realize the problem posing practice by ILE about simple arithmetical word problems.

In future work, we will refine functions in the ILE. For example, in the ILE, we will deal with not only Change-Problem, but also the other types of problems. And we will develop a function in which teachers can customize concepts provided for their students in their problem posing practice, because teachers hope to use concepts which are popular in their classroom. Then, we will evaluate the ILE again in order to investigate about the effect to learn arithmetic.

References

[1] Nakano，A.，T.Hirashima，A.Takeuchi，“Problem-Making Practice to Master Solution-Methods in Intelligent Learning Environment，” Proc of ICCE'99，pp891-898，1999.

[2] Barr,A., Beard, M., Atkinson, R.C., “The Computer as a Tutorial Laboratory: the Stanford BIP Project”. Int.J. Man-Machine Studies, Vol.8, pp.567-596, 1976

[3] MaCalla, G.I. “The Search for Adaptability, Flexibility, and Individualization:Approaches to Curriculum in Intelligent Tutoring ILEs”, In Jones, M. and Winne P.H.(Eds.), Adaptive Learning Environment: Foundation and Frontiers, Springer-Verlag, pp.91-122, 1990

[4] Hirashima, T., A.Kashihara, J.Toyoda : “Toward a Learning Environment Allowing Learner-Directed Problem Practice”. Proc. of ITS'96 Lecture Notes in Computer Science 1086, pp.466-474,1996

[5] Hirashima, T., A.Kashihara, J.Toyoda : “Providing Problem Explanation for ITS”, Proc. of ITS'92 Lecture Notes in Computer Science 608, pp.76-83, 1992

[6] Hirashima, T., T.Kohno, A.Kashihara, J.Toyoda : “Development of Support Facilities for Arithmetical Word Problem Exercises in Intelligent Tutoring ILE”. Electronics and Communications in Japan, pp.22-33, 1992

[7] Silver，E.A.，Mamona,J.，“Posing Mathematical Problems:An Exploratory Study，” Journal of Research in Mathematics Education，vol.27，No.3, pp293-309，1996.

[8] Lyn D. English，“Children's Problem Posing Within Formal and Informal Contexts，” Journal of Research in Mathematics Education，vol.29，No.1, pp83-106，1998.

[9] Ellerton, N. F., “Children's made up mathematics problems: A new perspective on talented mathematicians,” Educational Studies in Mathematics, 17, pp261-271, 1986.

[10] Silverman, F. L., Winograd, K., Strohauer, D. “Student-generated story problems. Arithmetic Teacher,” 39, pp6-12, 19992.

[11] Silver，E.A.，Cai,J.，“An Analysis of Arithmetic Problem Posing by Middle School Students，” Journal of Research in Mathematics Education，vol.27，No.5, pp521-539，1996.

[12] Silver, E.A, “On mathematical problem posing. For the Learning of Mathematics,” 14(1), 19-28

[13] Kintsch, W., Greeno, J.G., “Understanding and Solving Word Arithmetic Problems,” Psychological Review, vol.92, No.1, pp109-129, 1985.

A Method of Creating Counterexamples by Using Error-Based Simulation

Tomoya Horiguchi* and Tsukasa Hirashima**
**Faculty of Information Engineering, Kobe University of Mercantile Marine*
5-1-1, Fukaeminami, Higashinada, Kobe 658-0022 JAPAN
Phone: +81-78-431-6263, E-mail: horiguti@ti.kshosen.ac.jp
***Department of Artificial Intelligence, Kyushu Institute of Technology*
680-4, Kawatsu, Iizuka, Fukuoka 820-0067 JAPAN
Phone: +81-948-29-7618, E-mail: tsukasa@ai.kyutech.ac.jp

The method of creating counterexample by using educational simulation is proposed. Error-Based Simulation (EBS) is used for this purpose, which simulates a learner's erroneous equation in mechanics problem. A learner's error is visualized as unnatural motion of a physical object. In order for EBS to be effective as counterexample, the followings are essential: (1) A learner can recognize the difference of unnatural motion in EBS from natural one in correct simulation, and (2) EBS must provide a learner sufficient information to understand the cause of error and to reach correct understanding. The former has been studied in the authers' previous works. In this paper, the latter is discussed. To identify a learner's error, misconceptions are classified based on problem-solving model, and are linked to their appearance on a learner's answer (error-identification rules). Then, to indicate the cause of error by EBS, unnatural motions in EBS are classified and linked to the misconceptions which they suggest (error-visualization rules). These functions are realized as rule-base systems. The architecture of EBS management system, which judges a learner's error and generates the suitable EBS using these functions, is proposed.

Keywords: **counterexample, simulation, mechanics, error, student model, motion perception**

1 Introduction

It is well known that cognitive conflict promotes learning process. It often occurs when a learner encounters the fact which is contradictory to her/his idea. Cognitive conflict motivates a learner to reconsider her/his idea, and often causes conceptual change [Gagne 85, Fujii 97].

Counterexample is useful for creating cognitive conflict. It provides a case in which a learner's idea doesn't account for the fact, or her/his procedure doesn't produce the correct solution.

However, one must be careful in using counterexample, because a learner often ignores or refuses it. Even when she/he accepts the counterexample, she/he needs some kinds of help to reach correct understanding. Without any assistance, a learner often comes to an impasse, or makes ad hoc rules which explain the exception only. Therefore, in using counterexample, the followings are essential [Fukuoka & Suzuki 94, Nakajima 97]:

(1) Counterexample must be recognized to be meaningful and acceptable. When the difference is clear and reliable between counterexample and a learner's expectation, she/he easily accepts it and reconsiders her/his idea.

(2) Appropriate assistance must be provided to lead a learner to correct understanding. Counterexample must include sufficient information for this. It will be helpful to explicitly describe the distinguishing attributes of counterexample.

Error-based Simulation (EBS) is an educational simulation which provides a learner counterexample. It simulates an erroneous equation made by a learner in solving mechanics problem. In EBS, a learner's error often appears as unnatural motion of a physical object, which differs from her/his prediction (She/he can usually predict the correct motion).

The authors have developed the method of generating effective EBS mainly from the above viewpoint (1) [Hirashima et al. 98, Horiguchi et al. 99]. The condition on which a learner can recognize the difference between EBS and correct simulation was formulated (Criteria for Error-Visualization: CEV), and the mechanism to estimate the quality of difference was proposed, which considers both clarity and reliability of the difference.

However, though such an EBS motivated a learner by indicating the existence of errors, it was not sufficient to lead her/him to correct understanding. It didn't provide sufficient information for this.

Therefore, this paper proposes the method of managing EBS from the above viewpoint (2). EBS must justly indicate the cause of a learner's error, and suggest how to correct it. The followings are the requirements and approaches for this purpose.

(a) The function which identifies the cause of error behind a learner's erroneous equation or her/his handwriting diagram.
Approach: First, construct a problem-solving model of mechanics. Secondly, based on the model, classify the misconceptions which occur in problem-solving as causes of errors. Thirdly, classify the appearances of the misconceptions on a learner's equation or handwriting diagram. Lastly, appearances and causes of errors are linked together correspondingly. These are called *Error-Identification Rules*.

(b) The function which generates the EBS indicating the identified cause of error by unnatural motion of a physical object.
Approach: First, classify the unnatural motions in EBS. Then, link them to the corresponding causes of errors, considering what kind of unnaturalness suggests what kind of misconception. These are called *Criteria for Cause-of-Error-Visualization*. With these criteria, EBSs are estimated their effectiveness. When there is no EBS which is judged effective, other teaching methods will be considered.

2 Previous works in Error-Based Simulation

Before proceeding to the main topic of this paper, we outline the stream of study in Error-based Simulation, which may be helpful to clarify the present problem and the position of this paper.

***Stage 0* [Hirashima et al. 98 for summary]**

The fundamental idea of EBS is very simple. In mechanics problem, many learners feel difficulty in thinking by equations, so EBS maps their equations from mathematical world to physical world. It embodies a learner's error as unnatural motion of a physical object, which makes it much easier to recognize the error. Here, we assume that unnatural motion in EBS is differ from a learner's prediction, that is:

Precondition-1: A learner can predict the correct motion (in spite of her/his erroneous equation). This precondition is set through all stages of the research of EBS.

***Stage 1* [Hirashima et al. 98]**

Apparently, the key of this method is how a learner sees the difference between the unnatural motion in EBS and the predicted natural motion. At least, the difference must be noticed by a learner. When the difference of two motions is small, she/he may not notice it, or cannot judge which motion is correct (unfortunately, the ability of human vision is not so sensible). Therefore, we set the following assumption:

Assumption-1: EBS must satisfy CEV-1 and/or CEV-2 below to indicate the existence of error.

Condition for Error-Visualization 1 (CEV-1): There is a qualitative difference between the motion in EBS and the one in correct simulation, that is, the qualitative values of a physical object's velocity are different between them.

Condition for Error-Visualization 2 (CEV-2): There is a qualitative difference between the change of motion in EBS and the correct simulation, that is, the qualitative values of the derivative of a physical object's velocity are different between them.

Stage 2 **[Horiguchi et al. 99]**

When regarding EBS as counterexample, the viewpoints (1) and (2) in chapter 1 are important. We previously worked out how to estimate the effectiveness of EBS from the viewpoint (1). It is subdivided into two viewpoints: (1-1) how clear the error appears in EBS, and (1-2) how reliable the EBS is as counterexample.

From the viewpoint (1-1), the more CEVs the EBS satisfies, the more effective it is. In general, changing parameters of the mechanical system makes EBS satisfy more CEVs. For example, in Figure 2, the EBS based on erroneous equation m2a = T + μm2g (Figure 2d) satisfies CEV-1. (The qualitative value of relative velocity between two blocks is [+], while it is [0] in normal case.) But, when the mass of m2 increases, the EBS becomes to satisfy CEV-2 besides CEV-1. (The velocity of m2 increases, while it decsreses in normal case.) We categorized the methods of parameter-change and their influence on the clarity of errors.

However, from the viewpoint (1-2), such parameter-change harms the reliability of EBS, because a learner feels it factitious to change parameters too largely. The smaller parameter-change the EBS has (no change is the best), the more reliable it is. This discussion is summarized as follows.

Assumption-2: From the viewpoint of clarity, EBS should satisfy more CEVs.
Assumption-3: From the viewpoint of reliability, EBS should have less parameter-changes.

Stage 3 **[just this paper]**

In estimating the effectiveness of EBS, there is another, and important viewpoint: whether the EBS provides appropriate information for correcting the error, that is, the viewpoint (2) in chapter 1. Stage 0-2 have been mainly concerned with how to make a learner notice the error, while at this stage, our concern is how to make him correct the error.

For example, consider the erroneous equation m2a = T + μm2g (Figure 2d). From the viewpoint of reliability (1-2), the EBS shown in Figure 2c is generated. But it shows the string between two blocks shrinking, which may suggest something is wrong about tension of the string. It is misleading because the real cause of error is the friction of m2. In this case, the EBS in Figure 2e should be generated to indicate the cause of error. (It is generated when taking the viewpoint of reliability (1-2), but by accident.)

Of course, the viewpoints (1-1) and (1-2) are useful to impress on a learner the existence of error. However, in considering the error-correction, to generate EBS from the viewpoint (2) becomes necessary. It is the very topic of this paper.

3 Mechanism for Identifying the Cause of Errors

Now, we'll explain how to realize the functions described in chapter 1. The mechanism for identifying cause of errors is realized as follows:

1. to generate the correct solution by problem-solving model.
2. to specify the erroneous part of a learner's solution by comparing with the correct solution.
3. to identify the cause of error by applying the Error-Identification Rules, which link the appearance of erroneous part to its cause.

Here, a learner's solution means the equation and handwriting diagram made by her/him, from both of which the information about her/his problem-solving process is derived.

3.1 Problem-Solving Model

We deal with the mechanics problems of high school level, which ask a learner to set up equation of motion by using Newton's second law. The problem-solving process is divided into three steps [Robertson 90, Plötzner 94]:

step-1 to predict the motion of physical objects in the mechanical system qualitatively.
step-2 to enumerate the forces acting on each object.
step-3 to compose the enumerated forces and substitute them for the left side of formula F = ma.

Table 1. Force-Enumerating Rules (FERs) (abstract)

force	Rules for enumerating forces			
gravity	R0:	r0-c1	Object-1 has mass $m > 0$	
		→		
		r0-a1	Gravity F to Object-1	Qualitative
		r0-a2	Direction. vertically downward	Qualitative
		r0-a3	Magnitude. $F = mg$	Quantitative
friction	R3:	r3-c1	Object-1 and Object-2 are touching together ∧	
		r3-c2	coefficient of friction of touching surface $\mu > 0$ ∧	
		r3-c3	normal force N acting on touching surface ∧	
		r3-c4	Object-1 and Object-2 are moving oppositely along the tangent	
		→		
		r3-a1	friction Ff1 to Object-1	Qualitative
		r3-a2	friction Ff2 to Object-2	Qualitative
		r3-a3	Direction(Ff1). opposite to the velocity of Object-1	Qualitative
		r3-a4	Direction(Ff2). opposite to the velocity of Object-2	Qualitative
		r3-a5	Magnitude. $Ff1 = Ff2 = \mu N$	Quantitative

Set up the equation of the block of mass m, which pulled by external force F, moving on the floor of coefficient of friction μ.

erroneous equation: ma = F

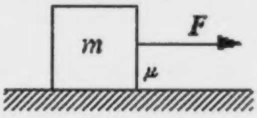


Figure 1. Example Problem-1

In step-1, a learner predicts the motion of objects in the system, and gives each object acceleration vector. Approproate axes are also set up. In step-2, she/he enumerates the forces which aren't given in problem description. Both qualitative knowledge (what kind of force acts in which direction?) and quantitative one (algebraic description of the magnitude of force) are used. In step-3, she/he decomposes/composes the enumerated forces along the axes, and substitute them for the formula F = ma.

In this paper, we don't model the error-occurring process in step-1, because it is presupposed that a learner correctly predicts the qualitative motion of objects in using EBS (Precondition-1 in chpter 2). We also omit the occurrence of error in step-3, which mostly concerns the knowledge of vector calculation.

Therefore, modeling step-2 is our central issue. Takeuchi and Otsuki (1997) considered that a learner constructs a model of causal structure of mechanical system, with which she/he infers the occurrence and propagation of forces. They formulated this process as a set of production rules. We modify them considering their qualitative/quantitative characteristics. A part of our model is shown in Table 1. The rules are called Force-Enumerating Rules (FERs).

3.2 Error-Identification Rules

In our model, a learner's errors are considered as the ones of FERs. The errors of FERs themselves and the ones in their application are included. In fact, these errors appear as the missing/extra/errors of the term of force in equation, or of the arrow of force in handwriting diagram. They are also linked to the strategies for correction.

For example, in Figure 1, the term of friction (-μmg) is missing in the erroneous equation. The cause of this error and its instruction are considered as follows:

1) A learner doesn't know the concept of friction itself, that is, doesn't know the rule R3 (Table 1).
Instruction: Re-teach the concept/definition of friction.
2) A learner is overlooking the preconditions of R3, that is, overlooking the fact that the block is touching the floor (r3-c1), or the fact the coefficient of friction is nonzero (r3-c2).
Instruction: Re-show the problem and indicate the corresponding part of the diagram.
3) A learner is missing the force which causes the friction, that is, missing the normal force (r3-c3).
Instruction: Proceed to the correcting strategy of normal force.
4) A learner doesn't think the block moves along the floor, that is, missing the relative velocity of them (r3-c4).
Instruction: This is the erorr of prediction of movement. So, out of the range of this paper. But, it may be useful to indicate the force which causes the block's motion.

Through such a consideration, the appearances of errors and their causes are classified as shown in Table 2. These are the Error-Identification Rules (EIRs), which are applied to the erroneous part of a learner's answer (specified by comparing with the correct solution), to identify the cause of error.

In Table 2, each error has its strategy for correction. Note that, it is not necessary to use EBS for every case. Of course, when other instruction method is more appropriate, it should be used. However, the aim of this paper is to clarify what kind of errors EBS is effective for, and how to estimate its effectiveness. For this purpose, we need to study the unnaturalness of physical objects' motion in simulation.

4 Criteria for Cause-of-Error Visualization

The identified error must be corrected. In this chapter, we formulate the criteria for judging whether an EBS is effective for the error. It means that EBS rightly indicates the cause of error and suggests the way of correction.

4.1 Motion and Forces

In EBS, it is the motion of physical objects (or their relationships) to be observed. Therefore, we classify the motions and connect them to the mechanical concepts they suggest.

Table 2. Error-Identification Rules (EIRs)

force	appearance	cause of errors	correcting strategy
external force/ gravity	missing extra error	missing knowledge of gravity (R0)	re-teach the concept/definition
		misunderstanding the problem (r0-c1)	re-show the problem and indicate the corresponding part
tension	missing	missing knowledge of tension (R1)	re-teach the concept/definition
		overlooking the string (r1-c1)	re-show the problem and indicate the corresponding part
		missing the force which causes tension (r1-c2)	proceed to the correcting strategy of that force
	extra	belief that string propagates tension (r1-c2)	re-show the problem and indicate the corresponding part
		overlooking that tension loosens string (r1-c2)	indicate that tension is extra
	error	error of the force which causes tension (r1-c2)	proceed to the correcting strategy of that force
		error of direction/magnitude (r1-a2/3)	indicate that direction/magnitude is erroneous
normal force	missing	missing knowledge of normal force (R2)	re-teach the concept/definition
		overlooking the touch/attachment (r2-c1)	re-show the problem and indicate the corresponding part
		missing the force which causes normal force (r2-c2)	proceed to the correcting strategy of that force
	extra	belief that normal force works (r2-c2)	indicate that normal force is extra
		extra of the force which causes normal force (r2-c2)	proceed to the correcting strategy of that force
	error	error of the force which causes normal force (r2-c2)	proceed to the correcting strategy of that force
		error of direction/magnitude (r2-a2/3)	indicate that direction/magnitude is erroneous
friction	missing	missing knowledge of friction (R3)	re-teach the concept/definition
		overlooking the touching together (r3-c1)	re-show the problem and indicate the corresponding part
		overlooking that coefficient of friction μ > 0 (r3-c2)	re-show the problem and indicate the corresponding part
		missing normal force (r3-c3)	proceed to the correcting strategy of normal force
		belief that normal force doesn't work (r3-c4)	indicate that friction is missing
	extra	missing that coefficient of friction μ = 0 (r3-c2)	re-show the problem and indicate the corresponding part
		extra of normal force (r3-c3)	proceed to the correcting strategy of normal force
		belief that normal force works (r3-c4)	indicate that friction is extra
		extra of the force which causes friction (r3-c4)	proceed to the correcting strategy of that force
	error	error of normal force (r3-c3)	proceed to the correcting strategy of normal force
		error of the force which causes friction (r3-c4)	proceed to the correcting strategy of that force
		error of direction/magnitude (r3-a3/4/5)	indicate that direction/magnitude is erroneous
propagating force	missing	missing knowledge of force propagation (R4/5)	re-teach the concept/definition
		overlooking the touching together (r4/5-c1)	re-show the problem and indicate the corresponding part
		missing of the force which causes force propagation (r4/5-c2)	proceed to the correcting strategy of that force
	extra	belief that force propagates (r4/5-c2)	indicate that propagating force is extra
		extra of the force which causes force propagation (r4/5-c2)	proceed to the correcting strategy of that force
	error	error of the force which causes force propagation (r4/5-c2)	proceed to the correcting strategy of that force
		error of direction/magnitude (r4/5-a2/3)	indicate that direction/magnitude is erroneous
other	extra	impetus	indicate that impetus is extra

BEST COPY AVAILABLE

How does a human perceive and recognize moving objects? Though it is well known that their figurative characteristics (figure, size, texture, etc.) and composition (position, direction, symmetry, etc.) have great influence on the arising images, it is difficult to generalize them because they much depend on the cultural factors. Therefore, we limit our target to the physical world of simulation, in which things are thought in the sense of mechanics.

When observing an object to move, a human feels its motive 'force' working. Of course, this kind of 'force' is of naive impression and doesn't always correspond to the real force. But it appeals to human's intuition so much more. Bliss & Ogborn (1992) classified such naive concepts of force according to the stages of child development. Based on their findings, we consider the relations between the motions in EBS and the forces they suggest.

4.2 Motion of a single object

A moving object arises the feeling of force working. (e.g. A falling down ball suggests gravity.) Therefore, the object moving unnaturally in EBS is supposed to suggest the erroneous force acting on it. (e.g. gravity, friction etc.) Unnatural motions of a single object are classified as follows:

(a) Directions of both velocity and acceleration are opposite to the ones of correct motion.
(b) Direction of only velocity is opposite to the one of correct motion.
(c) Direction of only acceleration is opposite to the ones of correct motion.
(d) Directions of both velocity and acceleration are same as the ones of correct motion.

Here, it is assumued that human can distinguish at most the qualitative difference of velosity or acceleration of an object in motion [Hirashima et al. 98, Horiguchi et al. 99].

For example, in case (a), when a learner observes an object moving in the opposite direction to her/his prediction (which is correct), she/he will recognize that the force is missing which acts in the predicted direction, or that the force is extra which acts in the present direction.

Table 3 shows the relations between unnatural motions and the errors they suggest. They are called Criteria for Cause-of-Error Visualization (CCEVs).

4.3 Relative Motion of two objects

Moving plural objects also arises the feeling of force working. We limit to two objects. When observing two objects moving together, the force maintaining their relative motion is felt. (e.g. A moving dolly pulling another one connected by string suggests tension.) Therefore, two objects relatively moving in unnatural manner in EBS are supposed to suggest the erroneous force interacting between them. (e.g. tention, normal force etc.) Unnatural relative motions of two objects are classified as follows:

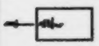
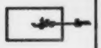
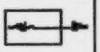
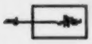
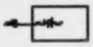
(e) Two objects are closing with each other, which are connected by string. (String shrinks.)
(f) Two objects are going away from each other, which are connected by string. (String stretchs.)
(g) Two objects are overlapping each other.
(h) Two objects are parting from each other, which are attached together.

For example, in case (g), when a learner observes such unnatural relative motion, she/he will recognize that the normal force is missing or too small which interacts between two objects.

Table 4 shows the relations between unnatural relative motions and their suggesting errors. They are also called Criteria for Cause-of-Error Visualization (CCEVs).

Note that, all of the motions in Table 3 and 4 have at least some kinds of qualitative difference from the correct motions. This is because, however precisely an EBS indicates the error, it isn't effective unless a learner recognizes it as 'unnatural.' The difference is judged with Criteria for Error-Visualization (CEVs) [Hirashima et al. 98, Horiguchi et al. 99].

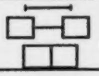
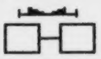
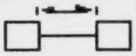
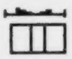
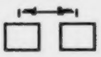
Table 3. Criteria for Cause-of-Error Visualization (CCEVs) (for single object)

motion		difference	suggesting errors
correct motion		–	–
motion (a)		velocity. opposite acceleration. opposite	missing of the force opposite to moving direction ⊙ extra of the force same as moving direction ⊙ larger of the force same as moving direction ○ smaller of the force opposite to moving direction ○
motion (b)		velocity. opposite acceleration. same	missing of the force same as moving direction △ missing of the force opposite to moving direction ⊙ extra of the force same as moving direction ⊙ extra of the force opposite to moving direction △ larger of the force same as moving direction ○ smaller of the force opposite to moving direction ○
motion (c)		velocity. same acceleration. opposite	missing of the force same as moving direction ⊙ extra of the force opposite to moving direction ⊙ smaller of the force same as moving direction ○ larger of the force opposite to moving direction ○
motion (d)		velocity. same acceleration. same	missing of the force opposite to moving direction △ extra of the force same as moving direction △

note 1. ⊙ . able to suggest the error by itself with great effect
○ . able to suggest the error by itself with small effect
△ . need to be modified some parameter(s) to suggest the error

note 2. The error of force in direction is divided into the missing of the force of correct direction and the extra of the force of incorrect direction.

Table 4. Criteria for Cause-of-Error Visualization (CCEVs) (for two objects)

motion		unnaturalness	suggesting errors
correct motion		constant distance	–
motion (e)		closing string shrinks	extra/larger of the tension ⊙ extra/larger of the propagating force ○
motion (f)		going away string stretches	missing/smaller of the tension ⊙ missing/smaller of the propagating force ○
motion (g)		overlapping	missing/smaller of the normal force ⊙ extra/larger of the normal force ○ missing/smaller of the propagating force ○ extra/larger of the propagating force ○
motion (h)		parting from each other	missing/smaller of the normal force ○ extra/larger of the normal force ⊙ missing/smaller of the propagating force ○ extra/larger of the propagating force ○

note 1. ⊙ . able to suggest the error by itself with great effect
○ . able to suggest the error by itself with small effect

note 2. The error of force in direction is divided into the missing of the force of correct direction and the extra of the force of incorrect direction.

5 Examples

In this chapter, we illustrate the process of identifying the cause of error and generating the EBS which indicates the error. The example problem is shown in Figure 2.

5.1 A Simple Case

First, the solution (correct equation and diagram: Figure 2a) is generated by problem-solver. Then, it is compared with a learner's answer (Figure 2b) to specify the erroneous part. In this case, it is the erroneous

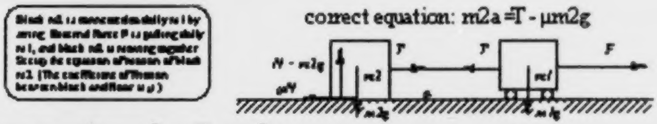
BEST COPY AVAILABLE

value (too large) of tension beside block m2. Secondly, EIRs (in Table 2) are applied to identify the cause of error. It is identified as the error of magnitude of tension. According to Table 2, the correcting strategy of this error is to indicate the fact. Then, CCEVs (in Table 3 and 4) are applied, to find that the motion (g) satisfies this demand.

Based on the erroneous equation of Figure 2b, the EBS shown in Figure 2c can be generated, in which block m2 moves faster than its normal case, consequently the string shrinks. This unnaturalness is equal to the one of motion (g). Therefore, this EBS is judged to satisfy the instructional demand, and shown to the learner.

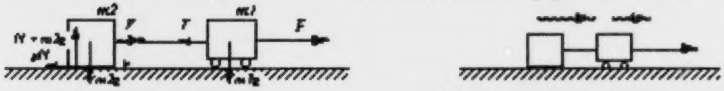
5.2 A Complicated Case

Consider the erroneous answer of a learner in Figure 2d. In this case, the erroneous part is the erroneous direction of friction acting on block m2. By EIRs, the cause of error is identified as the error of direction of friction, and the correcting strategy is to indicate the fact. Since the error of force in direction is divided into the missing of the force of correct direction and the extra of the force of incorrect direction (see note 2 of Table 3), the motions (a), (b), (d) satisfy this demand.



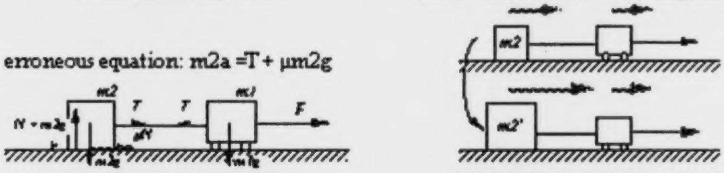
erroneous equation: m2a = F - μm2g

a. correct solution by problem-solver



b. erroneous answer-1

c. EBS which shows the string shrinking



d. erroneous answer-2

e. EBS with the perturbation of m2

Figure 2. Example Problem-2

Based on the erroneous equation of Figure 2d, however, it is impossible to generate the EBS which contains the motion (a) or (b). In addition, even when the EBS containing the motion (d) is generated (it is possible), it causes the unnatural relative motion (e), which indicates another error. In fact, the EBS, in which block m2 is closing to dolly m1 (the same as Figure 2c), strongly suggests the error of tension. This misleads a learner.

Therefore, in this case, the EBS must be modified to precisely indicate the identified error. Perturbing the mass of block m2 is a promising method. When the mass m2 increases, in EBS, the velocity of the block increases (Figure 2e). This is a strange change of motion. Observing this, a learner may think some physical amount is wrong which concerns the mass m2. She/he may notice the erroneous friction acting on block m2.

As for the EBS of Figure 2e, the difference from the correct simulation is not so much clear and reliable as the EBS of Figure 2c. Instead, it provides precise information for correcting the error, while the EBS of Figure 2c doesn't. In general, plural EBSs can be generated from one erroneous equation. The best should be chosen according to the purpose.

6 Concluding Remarks

In this paper, we proposed a method of creating effective counterexamples by using Error-Based Simulation. The effectiveness of EBS is judged mainly from the viewpoint whether it provides sufficient information to recognize the cause of error and correct it. The mechanism for identifying the cause of error and for

generating the EBS which satisfies the instructional demand was also proposed. We are now implementing the mechanism. The experiment to evaluate our method is planned.
Our future works are as follows:

1. *Cooperation with other instructional tools:* Of course, EBS isn't sufficient for all of the error correction in Table 2. It must be studied to use other instructional tools (textbook, normal simulator etc.), and to coordinate them with EBS.
2. *Refinement of the problem-solving model:* Our model for problem-solving is very simple, so the range of the error it covers is limited. We are going to refine the model, especially considering the process in which a learner qualitatively predicts the motion of mechanical system.
3. Consideration of conflict among CCEVs: As is noted in section 5.2, the effects of plural unnatural motions sometimes conflicts each other. One unnatural motion may invalidate the effect of other unnatural motion. Therefore, it is necessary to set some kind of preferences to CCEVs.

References

[1] Gagne, E.D. (1985). The Cognitive Psychology of School Learning, Little Brown and Company.

[2] Fujii, T. (1997). Mathematical Learning and Cognitive Conflict, In Japan Society of Mathematical Education (Ed.), Rethinking Lesson Organization in School Mathematics, Sangyotosho, Tokyo (in Japanese).

[3] Fukuoka, T. & Suzuki, K. (1994). A Case Study on Lower Secondary School Pupils' Conceptual Changes through Demonstration of Counterexamples: The Concept of Dynamics in the Case of Roller Coaster Models, Bulletin of Society of Japan Science Teaching, Vol.35, No.2, pp.21-32 (in Japanese).

[4] Nakajima, N. (1997). The Role of Counterevidence in Rule Revision: The effects of instructing metaknowledge concerning non adhocness of theory, Japanese Jour. of Educational Psychology, 45, pp.263-273 (in Japanese).

[5] Hirashima, T., Horiguchi, T., Kashihara, A. & Toyoda, J. (1998). Error-Visualization by Error-Based Simulation and Its Management, Int. Jour. of Artificial Intelligence in Education, Vol.9, pp.17-31.

[6] Horiguchi, T., Hirashima, T., Kashihara, A. & Toyoda, J. (1999). Error-Visualization by Error-Based Simulation Considering Its Effectiveness: Introducing Two Viewpoints, Proc. of AI-ED99, pp.421-428.

[7] Robertson, W.C. (1990). Detection of Cognitive Structure with Protocol Data: Predicting Performance on Physics Transfer Problems, Cognitive Science, 14, pp.253-280.

[8] Plötzner, R. (1994). The Integrative Use of Qualitative and Quantitative Knowledge in Physics Problem Solving, Frankfurt: Peter Lang Verlag.

[9] Takeuchi, A. & Otsuki, S. (1997). An Intelligent Tutor for Kinetic System Modeling, Proc. of AI-ED97, pp.340-346.

[10] Bliss, J. & Ogborn, J. (1992). Steps Towards the Formalisation of a Psycho-logic of Motion, In A. Tiberghien & H. Mandl (Eds.), Intelligent Learning Environments and Knowledge Acquisition in Physics, Heidelberg: Springer-Verlag.

Adaptive Programming Language Tutoring System on the Web

Jin-woo Choi and Chong-woo Woo
Department of Computer Science, Kookmin University
861-1 Chongnung-Dong, Sungbuk-ku, Seoul, Korea
Tel: +82-2-910-4795
Fax: +82-2-910-4799
Email: cwwoo@kmu.kookmin.ac.kr

Many of the web-based educational systems could not provide an individualized instruction or an interactive problem solving, since they are mostly built upon static hypertext. One possible approach to solve these problems could be adapting the existing proven techniques from the stand-alone Intelligent Tutoring System(ITS). Some recent web-based ITS researches show this efforts by employing the techniques selectively, and this needs to be studied further to support more effective web-based instruction. In this paper, we describe the design and the development of a Web based Adaptive programming Language Tutoring System(WALTS). The system is designed based on the ITS structure primarily, and it is adapting previous ITS techniques into the system successfully. Especially our focus is on the three levels of the instructional planning mechanism, which can generate lesson contents dynamically whenever it is requested. This way we do not need to crate all the lesson contents in HTML forms which must reside in the system in advance. In addition, the system has adapted CORBA structure to support the user more consistent and reliable performance. Together, the system behaves more adaptive and interactive, than the existing non-ITS based web systems. The test domain of the system is learning C programming language for the first year computer science student.

Keywords: **Web-based learning system, Intelligent Tutoring System, Instructional planning**

1 Introduction

Many recent web-based educational systems could not provide an individualized instruction or an interactive problem solving, since they are mostly built upon static hypertext. One possible approach to solve these problems could be adapting the existing techniques from the stand-alone ITS. Brusilovsky[2] states that some ITS techniques can be adapted into a web-based educational system, and divides the techniques into three distinctive parts, such as, automatic creation of curriculum, dynamic problem solving, and intelligent analysis of student model. However, most of the recent web-based ITS research show the efforts by employing the techniques selectively[2][3][4][7], such as adapting student modeling or problem solving capability at some level. Therefore, this needs to be studied further to enhance the overall capabilities of the system at the previous stand-alone ITS level. For instance, automatic generation of curriculum or lesson plan is necessary to provide a flexible instruction for each individual user.

In this paper, we describe the design and development of Web-based Adaptive programming Language Tutoring System(WALTS). The system is designed based on the ITS structure primarily, and it is adapting many techniques from the stand-alone ITS into the web-based systems. First, we designed the knowledge base using the object-oriented method in order to handle flexible management of object inheritance and tutorial strategies. Second, the student modeler can avoid the network traffic by designing the modeler stays in the server-side at the beginning of the session for maintaining necessary administration duties, and creates an individual student model in the client side. And the third important approach is having the instructional planning mechanism, which generates lesson contents dynamically for each individual user. This is important feature for moving towards the web-based system, because most web-based educational systems creates all the lesson contents in HTML forms in advance, and they must reside in the system physically. And then the user navigates the system for learning, such as in ELM-ART[2] or CALAT[4]. Intelligent navigation strategy might be one of the intelligent way of guiding the user to learn the material, but rather inefficient compare to the dynamic generation of lesson contents. WALTS only generates the necessary lesson contents whenever it is requested by the system, which can be another advantage. In addition, we have approached distributed architecture by employing CORBA(Common Object Request Broker Architecture) structure to support the user more consistent and reliable performance while the user using the system. The initial web-based educational systems are mostly developed by using the CGI(Common Gateway Interface) techniques, which often results in bottleneck problem when many users access the system at the same time. In this sense, our structure might avoid such a problem, and the system could also be easily updated when we need to revise some part of the system. Together, the system behaves more adaptive and interactive, than the existing non-ITS web-based educational systems. The test domain of the system is learning C programming language for the first year computer science student.

The rest of the paper is organized as follows. In section 2 we described a distributed infrastructure of WALTS system. Section 3 presents each components of the system and also some intelligent aspects of the system. We conclude the paper in section 4.

2 Distributed infrastructure

The previous web-based educational systems have been built as either a server-based architecture or a client-based architecture[6]. Each of them has some advantages and disadvantages. The server-side architecture mostly rely on CGI techniques, which has shown some problems of handling complex client/server communication because of its connectionless feature. Also client-side architecture needs to have all the plug-ins installed on client computer before using the system. Therefore the recent web-based applications tend to adapt CORBA or Java based distributed infrastructure. That is free from the connectionless or stateless problem, and also has some advantages of distributed system technology, such as message passing, RPC(Remote Procedure Call), and proprietary communication protocol. The client connects to the server using the HTTP protocol only for the initial connection, and after the downloading the specific mobile code application(for example, client side application, JavaScript, Java Applet, and etc), the client use the proprietary protocol(non-HTTP), so it does not communicates with web server, but communicates with proprietary server(non-Web server).

WALTS employed CORBA to adapt this kind of distributed infrastructure. The system is designed by HTTP server which takes care of user requests and responses, and CORBA-based server which performs the capabilities of the ITS. Also the system could be easily re-organized if we want to modify the structure later on [see figure 1]. In short, one of the major advantages of WALTS is that it can easily avoid the bottle-neck problem of CGI techniques, and also we believe that this style of architecture might be another best solution for building web-based client/server educational system.

3 Basic architecture of the system

The basic architecture of WALTS is designed by typical ITS structure primarily, including expert module, the student modeler module, and the instructional planning module.

3.1 The expert module

The expert module of the system consists of the object-oriented knowledge base, and the problem solver.

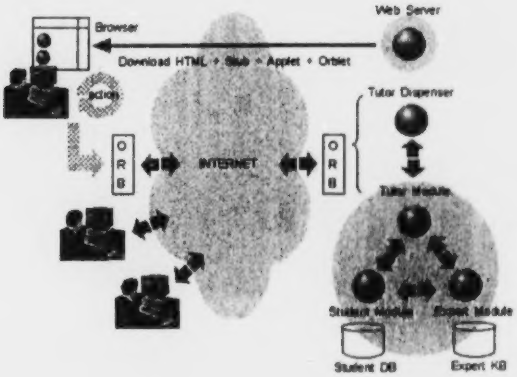


Figure1 The main architecture of the WALTS

Object Oriented-based knowledge base. First, we employed the frame knowledge representation techniques for the main knowledge base. Because the domain knowledge does not require any complex causal relationships, but rather it is consists of simple C language concepts. The object-oriented approach make it easy to modify the data type, can reduce the knowledge base reference by having slot values as member data, and can provide more flexibility for updating or manipulating tutoring strategy [5].

In this system we designed a frame with several meaningful slots, and each frame does not have to have the same number of slots, since the inference engine can get all the necessary informations due to the inheritance feature of the system. The 'type' slot can possess a concept, example, or quiz. The 'source' slot points to its superior frames. The 'Pfame' and 'CFrame' slot is necessary when we need to show the related nodes in linked list structure. The 'reference' slot may contain all the necessary frame names that are related to the current frame. This kind of slot structure is very common in every frame structure, and also important in object-oriented structure, because each frame can have common attributes and can generate an object of having its own attribute. Also, the system allows an abstract class, which plays the backbone of the system, and supports a hierarchical structure, and the definition of the method can be done only in the lower class [figure 2].

Frame Variable Declaration Quiz
[Source] Chapter1-3-1-1
[Type] Quiz
[Title] Variable declaration Quiz
[Template] Data Type | Variable | General Grammar
1 : Select the correct %type variable declaration
2 : ...
[PFrame] Variable Declaration
[CFrame] Null

Figure2 Variable declaration quiz frame

The Problem Solver. WALTS can generate a problem dynamically depending on the current topic. Since the planner knows what is being taught at the moment by generating a lesson unit, the tutor can decides whether it is 'teaching concept' or 'show example' or 'quiz'. At the moment, we have only three styles of lesson unit. If it is a 'teaching concept', the planner sends the lesson unit to the user in HTML form. If it is a 'quiz' type, then planner requests the problem solver to generate a question. The problem solver first creates a problem table by referring to the current lesson unit. The generating and solving a problem occurs at the same time, and the solver stores the correct answer. And then, it presents the generated questions to the user in appropriate HTML form through the HTML generator. This method can provide different styles of questions for different users even though they are accessing the same lesson unit, which can be another advantage of WALTS. Since the column name of the table is object's name, the planner can reply to the user's request, such as hint or help, by referencing this table.

BEST COPY AVAILABLE

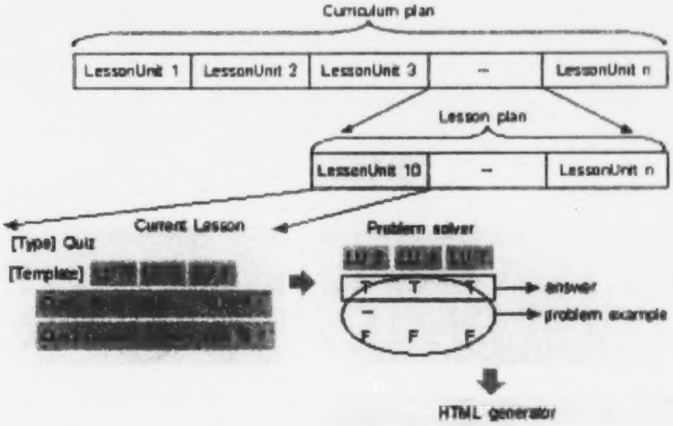

Curriculum plan
LessonUnit 1
LessonUnit 2
LessonUnit 3
--
LessonUnit n
Lesson plan
LessonUnit 10
--
LessonUnit n
Current Lesson
[Type] Quiz
[Template]
Problem solver
T T T
F F F
answer
problem example
HTML generator

Figure3 Creation of a quiz for 'variable declaration'

The strategy of asking user for answering quiz is multiple choices. So that we need to generate problems along with the appropriate multiple choice answers also. For instance, let us think about a simple quiz about asking user 'a data type'. A typical 'data type' is consists of three parts, for example, 'int x ;'.The 'int' is a data type integer, 'x' is a user-defined variable name, and ';' is needed for ending a sentence in C language. We are trying to generate this simple data type declaration statement sentence as follows. First, the data type 'template' slot consists of three parts as in [Figure 3]. Then we can generate eight different answers as in [Figure 5], since each one part of a statement can be correct or incorrect. And we can select some of them randomly including correct answer; the numbered answers are selected ones in the figure. And also we can obtain designated unit object's content as in [Figure 4]. The generated correct answer is stored in memory, and then later it is compared with the user's answer. For example, if the user selected number 2 as in [figure 5], we can analyze that the user does not know about reserved word. And the planner needs to revise the lesson plan to correct the misconceptions by giving special messages, such as hint or help, and then the planner re-organizes the lesson plan including 'reserved word' lesson unit. The [figure 6] shows a sample session of solving a generated quiz.

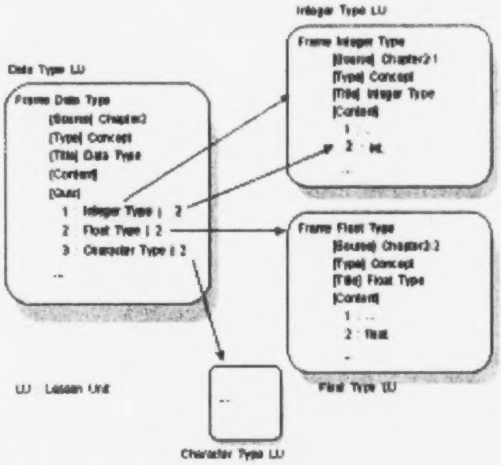
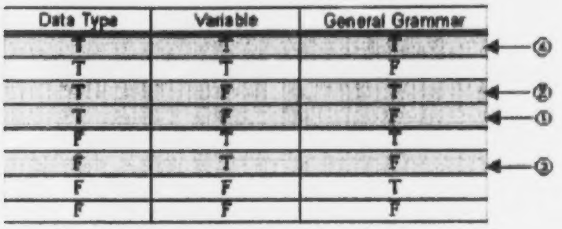

Integer Type LU
Frame Integer Type
[Source] Chapter2.1
[Type] Concept
[Title] Integer Type
[Content]
1 : ..
2 : int
...
Data Type LU
Frame Data Type
[Source] Chapter2
[Type] Concept
[Title] Data Type
[Content]
[Quiz]
1 : Integer Type | 2
2 : Float Type | 2
3 : Character Type | 2
...
Frame Float Type
[Source] Chapter2.2
[Type] Concept
[Title] Float Type
[Content]
1 : ...
2 : float
..
Float Type LU
LU : Lesson Unit
...
Character Type LU

Figure4 Data type unit frame

BEST COPY AVAILABLE

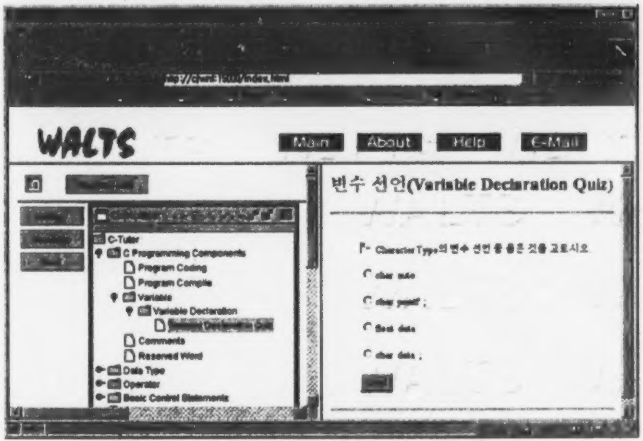


Data Type	Variable	General Grammar	
T	T	T	④
T	T	F	
T	F	T	②
T	F	F	①
F	T	T	
F	T	F	③
F	F	T	
F	F	F	

Figure5 Table for possible answers for data type quiz

3.2 The Instructional Planner

The most web-based educational systems built upon hypertext, which is hard to make hyperlink in every HTML pages, and also needs to have carefully designed navigation strategy[2]. And also all the lesson contents are built as HTML pages in advance, and must reside in the system physically. We believe that generating a lesson plan dynamically, for each individual user, is more efficient than the above approaches. Therefore, we adapted the traditional ITS instructional planning mechanism into the system. The instructional planning of the WALTS can be further divided into 3 steps, a curriculum planning, a lesson planning, and a delivery planning. The curriculum planning of WALTS generates a curriculum in tree structure; the curriculum planner extracts information from the knowledge base and creates a curriculum hierarchically in the order of prerequisites. Then the lesson planning sets up the lesson sequence within a single lesson unit. The role of delivery planning is limited to presenting the selected lesson content to the user.



변수 선언(Variable Declaration Quiz)

Figure6 An example of solving a generated quiz

Curriculum planning. The purpose of the curriculum planning is to provide a curriculum to the user, in other words, to provide an individualized optimal learning path to the user[1]. The generated curriculum is in the form of a tree structure. It is constructed by creating an initial node by referencing the value of the attributes in the lesson unit slot, and further expands the structure in the order of the way the student must learn, which will be accessed as linked list structure. The lesson unit of the system is organized according to some basic rule, such as the student must learn prerequisite concept first and the move to the next topic. So the curriculum is set up in the form of hierarchical and linear sequence.

Lesson Planning. The lesson planner generates a lesson plan by referencing the curriculum and the student model. The information from the student model shows the results from single lesson unit and based on this record, the planner sets up appropriate lesson plan for the student. When the student selects other learning path on purpose before the current lesson plan is finished, the system must decide what to do next, such as whether to store the current lesson plan and execute the user's request, and then resume the current plan or destroy the current plan and re-plan the whole sequence all over again. In that sense, WALTS uses re-planning strategy when the user wants to quit the current topic, and move to another learning path. Another

BEST COPY AVAILABLE

case of re-planning occurs when the student made an error on the selected quiz lesson unit. If the student made a mistake on this, the current lesson plan is suspended, and another new lesson plan is created to correct the student's error. After the remediation process is finished, the suspended plan will be resumed.

Delivery Planning. The lesson unit has been generated by lesson planner and needs to be delivered to the user. The possible delivery tactic in this domain could be "present concept, show example, give exercise, and etc". Of course if the system allows mixed-initiative control, the delivery planning needs to be more sophisticated in order to handle all the user request or questions. The delivery planning part of the WALTS is made of simple structure, and will be enhanced further in the next research.

The HTML generator. The very distinctive feature of the system is the HTML generator. This feature can be regarded as the interface part of the system. When the delivery planner decides the immediate unit lesson, the content of the lesson is converted into HTML form by the HTML generator. The HTML generator generates HTML pages according to the HTML2.0 protocol and inserts "next" or "previous" button in order to navigate adaptive learning path. But if the lesson unit contains some applet, the system directly searches the physical location and sends the URL to the student's browser without consulting HTML Generator. The [figure 7] describes the HTML generator sends two different results to two different users, since their learning background is different.

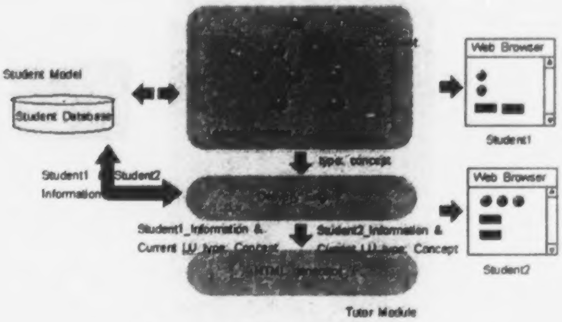


Figure7 The HTML generator

3.3 The student modeler

The strategy for building the student modeler is the simple overlay, which simply reflects user's learning process about current topic. And this should be enhanced by including the buggy information later on. But an important enhancement is that the student modeler of WALTS can avoid the unnecessary network traffic. For instance, if the system maintains the student model in the server-side, then whenever the user accesses the system the server needs to update the user's student model in the server. This may cause another bottleneck problem, and the most CGI-based systems still have this problem. Our approach on the student modeler is as follows. The server-side student modeler creates a table, and keeps all the necessary administrative informations on the server-side, such as initial student's ID, password, e-mail address, the access time[figure 8], which can be used for various administrator purposes. And the information regarding the student's learning process is stored in the student model[figure9], which is created in the client-side machine for each individual user whenever they logged on. The student model has several parameters that reflect the student's learning history, and each parameter has unique meanings. For example, the 'HelpCount', means how many times the user has been helped, and 'HintCount' means how many times the user has requested hints, and they can be updated only when the 'unit lesson' is quiz. The 'ReferenceCount' means the user is weak at the current unit lesson since the specific lesson has been accessed more often than other frames. The 'LessonLevel' stores information about how the level of the current topic, and the 'LessonType' means whether the current unit lesson is concept, example, or quiz, and so on.

BEST COPY AVAILABLE

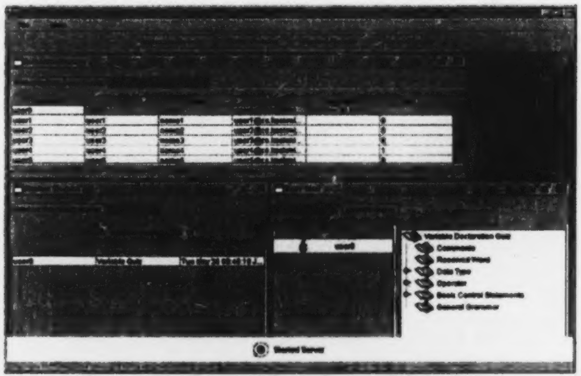
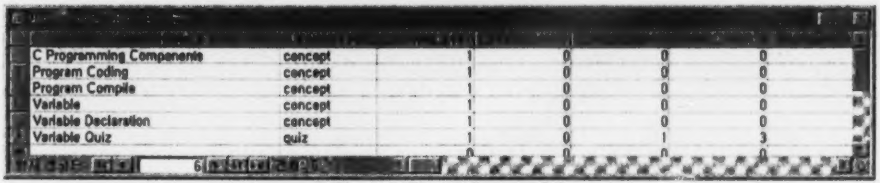


Figure8 The Server-side student modeler table



C Programming Componente	concept		1	0	0	0
Program Coding	concept		1	0	0	0
Program Compile	concept		1	0	0	0
Variable	concept		1	0	0	0
Variable Declaration	concept		1	0	0	0
Variable Quiz	quiz		1	0	1	3

Figure9 The student model

4 Conclusion

We have designed and implemented a web-based ITS, WALTS, which is a learning C programming language tutor aiming for the first year computer science students. The main goal of this paper is, first, the adaptation of the existing ITS techniques into the web platform. Therefore, we have designed and implemented the system based on the major ITS architecture, and this brings us several advantages over traditional HTML-based educational systems. First, the main knowledge base is created as an object-oriented concept, which can provide more flexibility for manipulating frame objects and tutoring strategy also. Second, we have generated a quiz dynamically by the problem solver and also can solve the problem. Third, we designed a student modeler that can avoid the network traffic in the minimal, by having the modeler in the server-side, and creates an individual student model in the client-side. Fourth, the instructional planner can generate an instructional plan dynamically, and this is another advancement of building web-based ITS, since the current web-based ITS research shows further work on this subject. Additional issue of the paper is that we designed the system as the distributed infrastructure using CORBA as backbone of the system. This structure solves the bottleneck problem of previous CGI dependent systems, and also gives some benefits of better performance and also gives flexibility in the case of further enhancement of the system.

References

[1] P. Brusilovsky. "Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies" http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/brusilovsky.html

[2] P. Brusilovsky, E. Schwarz, and G. Weber. "ELM-ART: An Intelligent Tutoring System on the WWW." The Proceedings of ITS'96, pp261-269, Springer, (1996).

[3] J. Kay and R.J. Kummerfeld "An Individualized Course for the C Programming Language." The Proceedings of 2nd International WWW Conference, Chicago, IL, pp17-20, (1994).

[4] K. Nakabayashi, M. Maruyama, Y. Koike, Y. Kato, H. Touhei, and Y. Fukuhara. "Architecture of an Intelligent Tutoring System on the WWW." The Proceedings of AI-ED'97, pp39-46, (1997).

[5] K. Nakabayashi, H. Touhei, S. Hitoshi, and Y. Fukuhara. "An Object-Oriented Architecture for a Web-based CAI System", The Proceedings of EDMEDIA'98, (1998).

BEST COPY AVAILABLE

[6] A. Silva, M. Silva, and J. Delgado. “A Survey of Web information Systems.” The Proceedings of WebNet’97, pp520-530, (1997).

[7] M. Stern, B. Woolf, and J. Kurose, "Intelligence on the Web?," The Proceedings of AI-ED'97, pp490-497, (1997).

Agent-Based Collaborative Learning Environment for Intelligent Tutoring System (ITS)

Teresita C. Limoanco and Raymund C. Sison
ccstcl@ccs.dlsu.edu.ph, ccsrcs@ccs.dlsu.edu.ph
College of Computer Studies, De La Salle University, Professional Schools Inc.
Manila, Philippines

This paper proposes a general architecture illustrating how students can learn through peer interaction in an interconnected environment. Three (3) predominant components comprise the architecture: ***the student model, the tutor model*** and ***a pedagogical agent*** known as *SPY*. The use of pedagogical agents is the essential part in the proposed architecture, in which information received from other students will be used as one of the tutoring strategies to assist students in learning. Collaborative/cooperative learning is achieved between students and the tutor, or among students, through pedagogical agent interaction. Moreover, the architecture supports a collaborative learning environment that helps improve students' comprehension.

Keywords: **collaboration, collaborative learning, agents, intelligent tutoring system**

1 Introduction

With the advancement in technology, computers have become essential tools in developing systems that cater to the different needs of users. Currently, many works have been done in the field of education. Systems known as *intelligent tutoring systems (ITSs)* were developed to teach students on specific topics, test their knowledge by giving exercises, and provide remediation on topics students did not perform well. An *intelligent tutoring system* is a computer program for educational support that can diagnose problems of individual learners. Such diagnostic capability enables it to adapt instruction or remediation to the needs of individuals [5]. Currently, the state of ITSs is focussed on one-on-one learning instruction. Specifically the kind of learning modality used is centered on *learning by being told* [2]. However, in reality, students can also learn through interactions with his/her peers or work in a team (or a group). The information students receive from his peers can help improve his comprehension on the topics at hand. A new learning paradigm has emerged aiming on this area and this new learning paradigm is known as *collaborative learning*. Collaborative learning emphasizes on how students function in a group and how the students' interaction with his peers or work in a team can help improve students' learning. This can be seen as either gaining new knowledge or verifying the correctness of what the students had learned so far.

Meanwhile, one of the major issues in Distributed Artificial Intelligence involves multi-agency. The agents in a multi-agent system are designed to solve a kind of problem. This is based on the fact that agents are autonomous and can recognize their own existence and the existence of other agents. Agents help each other in order to achieve a common purpose within a certain environment. Agents can assist each other by sharing the computational load for the execution of subtasks of the overall problem, or through sharing of partial results that are based on somewhat different perspectives of problem solving on the overall problem. Moreover, this form of cooperation addresses the nature of communication between cooperating agents.[1,6] Due to the social ability[1] and proactivity[2] of agents, many research, and works whether related on education (i.e., ITSs) or interface learning have been done with the incorporation of agents. Some works include defining software agents to analyze the collaboration in a virtual classroom [3]. [2] proposed a system that is a CSCW environment with artificial agents assisting students in their learning tasks. Furthermore, it uses a tutor agent partially replacing the human teacher. [4], on the other hand, showed that the tutoring knowledge in ITS can be designed and organized as a team of interacting pedagogical agents. These agents communicate with the student depending on the tutoring function they fulfill. Some tutoring functions include domain presentation, domain assessment, problem solving type of exercises, topic selection, problem-step solving, domain explanation, and the like.

---

[1] Ability of agents to interact with other agents and human agents through some kind of an agent communication language. [3]

[2] Agents do not simply act in response to its environment, but they are able to exhibit goal-directed behavior by taking the initiative. [3]

This paper presents a new approach to collaborative learning using agents. The collaboration of agents is seen as sharing of information in the environment. The main thesis is that information received from other agents can be used as one of the tutoring strategies of other students learning in the network. The paper is organized as follows: Section 2 describes the general proposed collaborative learning architecture, the different components associated and their interrelations. Section 3 presents how artificial agents can support the learner and describes the kind of learning strategy each agent should posses in the system. The last section summarizes some issues that need to be considered in the proposed architecture.

2 Collaborative Learning Environment

The learning comprehension of a student on certain topics can be improved if the student is allowed to interact with his peers and not only with the tutor. This is because the way a student understands a topic can be applied as the same approach for other students who have difficulty comprehending the same topic. For example, two students (Student A and Student B) are studying at the same time on the same topic. They may be physically present but in different places. Student A is able to understand the topic well, but B is having difficulty with the topic. Instead of leaving the topic without fully understanding it (for the current topic may have an effect on the succeeding topics), B can either collaborate with his tutor or with A. Since B's tutor may use the same approach in explaining the topic as he did earlier, B "collaborates" with A. The collaboration may be in the form of using the same tutoring strategy used for A. With this, it is essential to develop ITSs that allows students' interaction that goes beyond the student-tutor relationship.

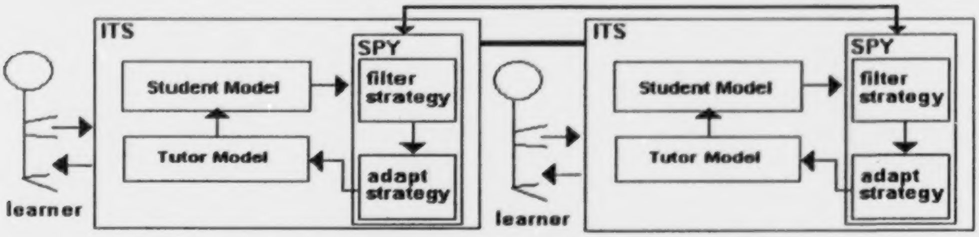


Figure 1: Agent-based Collaborative Learning Architecture

Figure 1 shows the proposed agent-based collaborative learning environment of an ITS system. The proposed architecture intends to illustrate collaboration that is not limited to student-tutor relationship but allowing students to interact directly or indirectly with his peers. In addition, this is done through interaction of the pedagogical agent SPY with the other agents of ITSs in the environment. It is assumed in the architecture that there can be several ITSs in the environment for a given domain. Though there may be the same set of tutoring strategies for the ITSs, it is possible that different tutoring strategies are used for the same topic. For instance, both student A and student B are currently studying lesson 1. However, the tutoring strategy used for A is presentation of lectures with illustrations, while B uses simple presentation of lectures. Furthermore, these ITSs are interconnected in a reliable network. The architecture can be implemented in either an Internet or Intranet infrastructure. Thus, it is good and useful for open and distance learning education.

There are three (3) main components in the architecture and these are the *tutor model, the student model* and *a pedagogical agent* known as *SPY*. Each of these is discussed below:

- ***Student Model:*** This module contains information about students' profiles and behaviors. Such information involves what the student has learned so far, has not learned, will be about to learn, and the possible misconceptions and their explanations on topics presented during the learning activity. Furthermore, the student model keeps track of the performance level of the students.
- ***Tutor Model:*** This module is responsible for the delivery of topics to students. Moreover, the tutor model also determines and delivers exercises to be solved by students. It is inherent in the tutor model that when presenting the exercise it considers student's level.

- ***SPY***: Each student is assigned an autonomous agent in the learning environment. This agent is responsible for gathering information such as the tutoring strategy used, the topic where the tutoring strategy is applied and the performance of students during his interaction with the ITS. Furthermore, SPY collaborates with other agents in the environment, with or without the presence of the student and the tutor. This means that SPY persists even if the student is not using the ITS. The information gathered will be used to determine the appropriate tutoring strategy for a particular topic for a student.

Specifically, the student interacts with the ITS in order to learn new concepts or to verify the correctness of what he has learned so far. During the learning activity, the student model monitors the performance of the student, keeping track of what the student has learned and is currently learning and his performance during the learning session. Any misconceptions the student may have are also being monitored. The tutor model presents topics according to the level of understanding of the student. The same approach is done when presenting exercises to students. The student's level of understanding can be determined from the student model. Information stored in the student model is then passed to the agent SPY, which in turn uses the information to determine which students have similar profile as its human student (i.e. relatively same performance, same learning style and relatively of the same level). From this interaction, the agent will gather data such as how other students were able to solve similar problems and how the topics were presented to them (i.e. tutoring strategy used). Moreover, SPY will keep track of the topics where the tutoring strategy was used and the student's performance. The rational behind this is that it is possible for SPY to determine in advance the tutoring strategy used for topics that are not yet presented to its human student. Consequently, this will allow the tutor model to adapt tutoring strategies depending on the status of its student.

The architecture illustrates two (2) forms of collaboration. The first is where the student collaborates with other students through communication medium and tools provided by the environment. These tools include chat, exchanging of emails/messages, discussion groups, newsgroups and the like. In this way, students can get actual explanations of how their peers understood the topics, concepts and solutions of problems presented during the learning activity. The second form of collaboration is where agents interact with other agents in the environment. Such interaction is abstracted from the students. The collaboration seen here is the sharing of information among agents about the students they are associated with.

To illustrate the second form of collaboration in the proposed learning architecture, consider this example: Students may or may not use the ITSs at the same time and study the same concepts or topics. In either case, the respective agents of each ITS in the network will still have to communicate and obtain information from other agents. Assuming there are two students, A and B who are present in the network and are interacting with their respective ITSs. Student A is currently studying topic 1 and student B is studying topic 2. Any interactions both students do during their learning activity are being monitored by their respective student models. While A is studying topic 1, his corresponding SPY agent is interacting with other agents (including the agent SPY assigned to student B) keeping track of topics other students have learned or is learning, the kind of tutoring strategy used, and the students' performance. By the time A is about to study topic 2, the tutor model A will adapt the tutoring strategy used for student B. This is with the assumption that the tutoring strategy used in B is effective (i.e., student B was able to understand topic 2 well, as this can be seen by his performance on that topic). If the adapted tutoring strategy is not appropriate to A (i.e., the student did not perform well in the corresponding exercises) another tutoring strategy will be used or by default will use the tutoring strategy of the tutor model.

It can be seen from the second form of collaboration that adaptation of tutoring strategies exists. Moreover, it is possible for the tutor model to change the current tutoring strategy used depending on the performance of the student during his interaction. The first form of collaboration allows students to directly apply what he/she has learned from the interaction.

3 The Pedagogical agent SPY

The agent SPY is introduced in order to allow students to share to their peers what they have learned and how they have learned the concepts. Specifically, SPY continuously communicates with other agents in the network keeping track of the approach or strategy used by other tutor models in teaching the concepts. Once information is gathered, SPY will perform 2 main operations: (1) filtering the strategies acquired from other agents and (2) transform the acquired strategy into a representation that can be adapted by the tutor model. The filtering of strategies is done in order to choose the appropriate strategy that can be applied to the current topic or concepts the student is studying. During the agent interaction, each agent can gather more than one kind of tutoring strategy possibly for the same topic or concept. These strategies can be arranged in many forms or classifications. For instance, strategies can be arranged according to its effectiveness based on students' performance. This means the strategy with a student receiving the highest score will be adapted and used by tutor models of other students. If such strategy is not applicable to the current student, then the next highest scored student's tutoring strategy is applied. It is also possible for the agent SPY during the filtering of strategies to combine similar strategies into one strategy. Or better yet, arrange the different tutoring strategies according to the topics or concepts that have been presented or learned by students.

Once a strategy is selected, it will be transformed into a representation recognizable by the tutor model. This is done by following an adaption algorithm. The adaption algorithm should be flexible such that it can adjust to and apply any kind of strategies. However, it is possible that the adapted tutoring strategy is the same as the intended tutoring strategy of the tutor model. In this case, this will serve as a "confirmation" to the tutor model that the pre-planned tutor strategy is effective to its human student.

The objective of introducing a pedagogical agent in designing an ITS is to support the student in learning by adapting different approaches in presenting the topics. This is with the hope of improving students' learning comprehension. Furthermore, SPY assists the tutor model as to what kind of teaching strategy to use for certain concepts. In addition to being adaptive and reactive to the needs of students, SPY agents are proactive and goal-oriented in the sense that they act in the environment through its initiative.

4 Conclusion

In this paper, a proposed agent-based collaborative learning architecture for designing an ITS is presented. The architecture is general and at the moment no implementation has been made. The architecture has shown how a pedagogical agent can be used to model collaborative learning. There are three (3) main components in the proposed architecture and the predominant component is the inclusion of a pedagogical agent known as *SPY*. The agent *SPY* is introduced to assist the tutor model in determining which teaching strategy it will be used in presenting topics/ concepts to students. This also includes the presentation of exercises and possible remediation on topics students are having difficulty with.

This paper also showed a different form of collaboration that is not the same as the usual collaboration or teamwork that is seen in reality. The paper proposes a form of collaboration in which there is a sharing of information among the agents and students in the environment.

Certainly, much progress has to be made towards reaching the complete architecture in reality. Particularly, in-depth study and implementation of the said proposed architecture is needed to see if the architecture can provide improvement in student's learning comprehension. Moreover, there are several issues on the proposed architecture that needs to be studied carefully. Some issues include the learning capability of the tutor model to adapt new tutoring strategies from the SPY agent; representation and storage of strategies (i.e., how can strategies be represented in the form of rules and how to store them in each agent); filtering of strategies (i.e., how to determine which of the acquired strategies are useful and appropriate to the current performance of the students). In addition, a criterion needs to be defined on how to determine students with similar profile.

References

[1] Cammarata, Stephanie, McArthur, David, Steeb, Randall. "Strategies of Cooperation in Distributed Problem Solving". Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, Inc. San Mateo, California. pages 102-105. (1983)

[2] Florea, Adina Magda. "An Agent-based Collaborative Learning System". Advanced Research in Computers and Communications in Education, G. Cumming et al (eds), IOS Press, 1999. International Conference on Computers in Education '99, pages 161-164. (1999)

[3] Jaques, Patricia Augustin, de Oliveira, Flavio Moreira. "Software Agents for Analysis of Collaboration in a Virtual Classroom". Enable 99: Enabling Network-based Learning, Espoo, Finland, June 2-5. (1999).

[4] Lelouche, Ruddy. "An Interacting Team of Pedagogical Agents for Intelligent Tutoring of One Student". Advanced Research in Computers and Communications in Education. G. Cumming e al. (Eds). IOS Press. International Conference on Computers in Education 1999. pages 157-160. (1999)

[5] Sison, Raymund, Numao, Masayuki and Shimura, Masamichi. "Multistrategy Discovery and Detection of Novice Programmer Errors". Machine Learning, 38, 157-180. Kluwer Academic Publishers. (2000).

[6] Smith, Reid, Davis, Randall. "Frameworks for Cooperation in Distributed Problem Solving". Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, Inc. San Mateo, California. pages 61-69. (1981)

An Educational System that can Visualize Behavior of Programs on the Domain World

Taketoshi Ando*, Yoshinobu Kawada*(†), Ryoji Itoh*,
Tatsuhiro Konishi*, Yukihiro Itoh*
* *Department of Computer Science, Shizuoka University.*
Johoku3-5-1, Hamamatsu, Shizuoka, 432-8011, Japan.
Tel: +81 - 53 - 478 – 1474• • Fax: +81 - 53 - 478 – 1499
E-mail: s5005@cs.inf.shizuoka.ac.jp

In this paper, we discuss extension of our educational system that gives domain-oriented-explanations of programs. A programmer solves problems on a world where elements necessary to describe the problems and the solving processes of them (objects in the problems, relation among the objects, and so on) are represented. We call such a world 'the domain world' of the problem. Our system has a domain world model and simulates a target program on the model, to understand behavior of the program. By analyzing the result of understanding, it generates an explanation. Outputs of our original system are only verbal explanations. However, when the system explains by using only sentences, some learners cannot get a concrete image of behavior of the program. Therefore, we are trying to add a facility of generating explanations by using animations (visual explanations) to the system. Our extended system can generate both visual and verbal explanations (bimodal-explanations) in various abstraction levels. We discuss the method of generating bimodal-explanations from the result of simulation.

Keywords: **Intelligent Tutoring System, Programming Education, Algorithm Animation System, Bimodal-explanation**

1 Introduction

The purpose of our research is to construct an educational system that helps novice programming learners by explaining domain-oriented-functions of programs. We take Pascal as our target programming language.

Programming is generally carried out in the following process.

Step1. A programmer understands a problem that must be solved.
Step2. He considers the solving process of the problem on a world where the problem is present. We call such a world 'the domain world' of the problem. For example, when he considers a solving process of sorting, he imagines a world in which he pays attention to numerical order such as greater and lesser (we call this world the world of greater and lesser).
Step3. He implements the algorithm: selects data structures suitable to represent the domain world and translates the algorithm into a programming language.

Usually, relatively simple problems are set in novice class of programming. So it is rare that learners fail in the step 1. But they tend to confuse because they cannot distinguish between step 2 and 3. So many novice programmers cannot find whether the causes of bugs are hidden in the algorithm or in their implementation. On the basis of this idea, we proposed an educational system explaining programs using vocabularies on a domain world[2][5][6][7]. Difference of our system from existing educational systems of programming [1][8] is that the purpose of our system isn't pointing out bugs in learner's programs, but rather helping learners find bugs by themselves. Our system helps learners in the following way:

- To help learners to understand sample programs given by a teacher by explaining them.
- To help learners to find and fix bugs in their own programs by explaining the faulty behavior of them.

Our previous system outputs sentences using vocabularies on a domain world as the explanation. However, when the system explains by using only sentences, some learners cannot get a concrete image of behavior of the

---

(†) *Presently with System Integration Group, VICTOKAI, LTD.*

program. If animations of the behavior of programs are shown with the sentences, learners can easily understand their algorithms. Therefore, we realize the ability to generate animations (visual explanations) that show behaviors of the target programs. In this paper, we discuss the way to generate visual explanations for programs in the domain world of greater and lesser.

Existing algorithm animation systems can be classified into two types: The first one is a system such as courseware editors embody particular commands to target programs in order to generate visual explanations, like Zeus[3] and TANGO[9] system. So, this type of systems can generate visual explanations of high quality by using concrete objects on the domain world. For example, a length of bar is used to concrete values of variables on the visual explanation of XTANGO system. The second type of system doesn't need embodying particular command to target programs, like UWPI[4] and tracers. However, this type of systems cannot generate any visual explanation using concrete objects on the domain world. They can only generate visual explanations showing structure of data and changes of contents of variables. Our system can generate a visual explanation using concrete objects on the domain world without embodying any special commands to programs. It generates visual explanations on the basis of the result of "simulation based program understanding[5]". So it can accept buggy programs and generate visual explanations of buggy behaviors of the programs. Moreover, it can also generate verbal explanations on the basis of the result of program understanding.

In the next section, we illustrate an overview of our previous system. In section 3, we point out some functions necessary to generate an effective explanation by using both verbal explanation and visual explanation (a bimodal-explanation). In section 4, we describe the method of constructing the bimodal-explanation system. Then, we show examples of bimodal-explanations by our system.

2 Our Previous Work

2.1 Overview of our previous system

Our previous system is composed of the static analyzer, the simulation based analytical unit and the explanation unit (Figure 1). In this paper, we omit detail of the system (For further details, please see our previous papers[2][5][6][7]). The static analyzer parses target programs and analyzes information necessary for the simulation such as data flow. The simulator simulates target programs, and the observer observes the world model while simulation, and recognizes some important characteristics of data or patterns of structured data. The explanation generator generates verbal explanations of target programs.

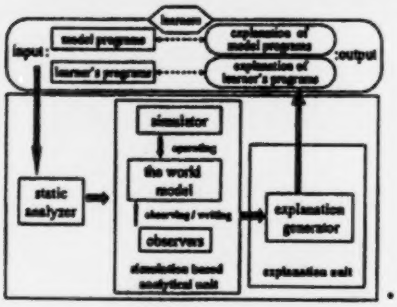

learners
input:
model programs
learner's programs
explanation of model programs
explanation of learner's programs
:output
simulator
operating
the world model
observing / writing
observers
static analyzer
explanation generator
simulation based analytical unit
explanation unit

Figure 1: Configuration of our system

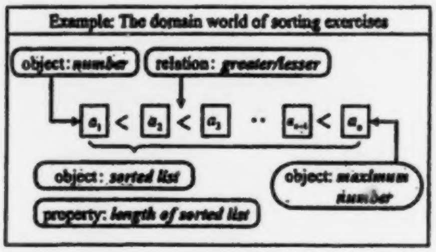

Example: The domain world of sorting exercises
object: number
relation: greater/lesser
a_1 < a_2 < a_3 ·· a_{n-1} < a_n
object: sorted list
object: maximum number
property: length of sorted list

Figure 2: An example of domain world

2.2 Domain world models

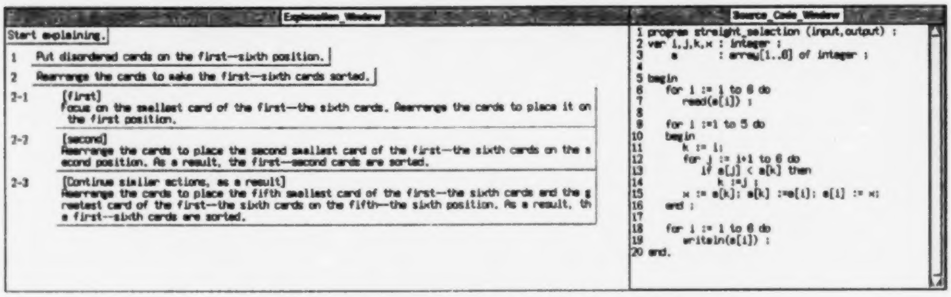
We examine programming exercises and classify them into 15 types. We prepare domain world models designed for each type of exercises [2].

A domain world model consists of four types of elements called 'object', 'property', 'relation among objects' and 'change'. For example, Figure 2 shows the domain world model of greater and lesser as an example. In order to recognize specified characteristic or patterns in the domain world, our system has daemon units called "observer" which are burnt when they are observed. In the Figure 2, the object *'maximum number'*, *'sorted list'* and property *'length of sorted list'* are recognized by observers. There are some cases that some observers take outputs of the other observers as their inputs. Then the outputs of observers make hierarchy. When a result of observation is output on the basis of a result of another observer, the former has larger grain-size than the latter and implies the fact corresponding to the latter.

BEST COPY AVAILABLE

2.3 Generation of a verbal explanation

The explanation unit generates verbal explanations of the target program by using results of simulation and outputs of observers. The results of observations have a hierarchical structure, as mentioned above. The system generates a hierarchical verbal explanation by using the hierarchical structure (it also uses syntactical structures of programs). In other words, the system notices the largest grain-sized result of the observation firstly, in order to generate the verbal explanation. Secondly, if learners request the detailed verbal explanations, the system generates the explanation using results of observation having smaller grain size. Figure 3 shows the example of verbal explanations generated by our system. It illustrates the verbal explanation of behavior of a sorting program on the domain world of greater and lesser. The indentation in the figure means that *behavior 1* and *behavior 2* are executed sequentially and that *behavior 2* is equivalent to the sequence of *behavior 2-1, behavior 2-2,* and *behavior 2-3*. Each Behavior is implemented by a single statement or a sequence of statements. When a verbal explanation for a behavior implemented by a sequence of statements is clicked, more detailed verbal explanations showing the way to implement the behavior are displayed.



Start explaining.

1 Put disordered cards on the first—sixth position.

2 Rearrange the cards to make the first—sixth cards sorted.

2-1 [first]
Focus on the smallest card of the first—the sixth cards. Rearrange the cards to place it on the first position.

2-2 [second]
Rearrange the cards to place the second smallest card of the first—the sixth cards on the second position. As a result, the first—second cards are sorted.

2-3 [Continue similar actions, as a result]
Rearrange the cards to place the fifth smallest card of the first—the sixth cards and the greatest card of the first—the sixth cards on the fifth—the sixth position. As a result, the first--sixth cards are sorted.

```
program straight_selection (input,output) ;
var i,j,k,x : integer ;
    a       : array[1..6] of integer ;

begin
   for i := 1 to 6 do
      read(a[i]) ;

   for i :=1 to 5 do
   begin
      k := i;
      for j := i+1 to 6 do
         if a[j] < a[k] then
            k :=j ;
      x := a[k]; a[k] :=a[i]; a[i] := x;
   end ;

   for i := 1 to 6 do
      writeln(a[i]) ;
end.
```

Figure 3: An example of the verbal explanation of a target program

Next, we show procedures for generating the hierarchical verbal explanation like Figure 3. An input is a result of simulation of some statements (For further details, please see our previous paper[7]).

(1). The case that a certain behavior is implemented by a sequence of statements.
The system observes differences of the states of the domain world model before and after execution of the sequence of statements. According to these differences, the explanation unit selects a template and generates verbal explanations for the statement. Now we show an example of a template.

- The case that extension of the sorted list is observed.
  The differences are composed of the following three elements.
    Object(s) recognized at the state before execution of some statements: a sorted list
    Object(s) recognized after execution of the statements : an extended sorted list
    Recognized changes of states of objects : an extension of the sorted list

A template for the extension of the length of a sorted list is applied (Please see Figure 3).

Template: "Rearrange the [Type of added object] to place [An added object] on [The position of the insertion] position. As a result, [A sorted list at the after state] [Type of inserted objects] are sorted."

"[]" means a procedure which generates a certain pattern of string.••

[An added object]
: A procedure that generates a noun phrase expressing the new object added to the sorted list.
[A sorted list at the after state]
: A procedure that generates a noun phrase expressing the range of the sorted list at the after states.

(2). The case that a certain behavior is implemented by a single statement.
The explanation unit calls each procedure corresponding to types of the statement. The procedures are defined for each structure of the program like sequential structures, selective structures, iterative structures, an assignment statement, a statement for input, and a statement for output. Same as the case (1), templates are prepared for each structure of the program. For example, we show a template of 'if' statement.

Template: "if [explanations of the conditional clause], [explanations of the 'then' clause]
(otherwise [explanations of the 'else' clause]) "

[explanations of the conditional clause]
: The procedure that explains the conditional statement of 'if' statement.
[explanations of the 'then' clause]
: Apply the procedure for generating the verbal explanation to the clause recursively.
[explanations of the 'else' clause]

: Apply the procedure for generating the verbal explanation to the clause recursively.

Thus, the system can generate hierarchical verbal explanations. When a verbal explanation generated by the procedure (1) is shown and a learner requests more detailed explanation, the system tries to apply the procedure (1) recursively to make such an explanation. If it cannot generate any explanation, it applies the procedure (2).

3 Functions necessary to generate an effective bimodal-explanation

In order to construct a system generating effective visual explanations, we have to consider what visual explanation is effective for learners to understand an algorithm or behavior of a target program. By designing mock up visual explanations repeatedly, we find that the effective visual explanation has following three facilities.

(1) The facility to generate visual explanations with various grain-sizes.
When learners learn programming by using a system explaining behaviors of programs, they need various grain-sized explanations. For example, when a learner wants to grasp algorithm roughly, a large grain-sized explanation would be effective. On the other hand, when he wants to understand a precise method of implementation, smaller grain-sized explanations are effective. Moreover, when he wants to diagnose his own program at a glance, he needs the largest grain-sized explanation. When he wants to find buggy codes, he needs smaller ones. In order to generate such various grain-sized visual explanations, the system should be able to :

- regard a sequence of statements as a blackbox and generate a visual explanation showing its function.
- generate a visual explanation showing a function of each statement sequentially.

(2) The facility to explain a function of a program by using both animations and verbal texts.
If a system shows only visual explanations, learners sometimes cannot understand behavior of target programs clearly, because such learners cannot understand what phenomena are essential. Thus, it is necessary for our system to have the facility to generate verbal explanations showing a major phenomenon of each step of visual explanations. Thus our system should have a facility of generating combination of verbal explanations and visual ones (bimodal-explanations).

(3) The facility to generate explanations on the total effect of a sequence of statements.
Generally, a task is achieved by a sequence of statements, and each sub-task is achieved by each sub-sequence of the statements. When the system shows a sequence of explanations each of which has a certain grain-size corresponding to a sub-task, a learner sometimes cannot find the fact that the task has been achieved. In order to prevent learners from such misunderstanding, the system should show them a verbal explanation remarking the fact.

4 Methods to realize the functions to generate bimodal-explanations

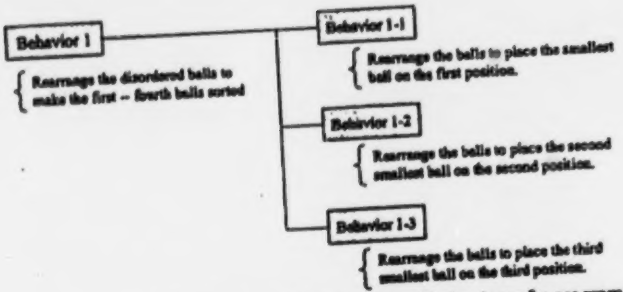
4.1 Basic ideas

(1) The method of generating visual explanations on various grain-size.
As we describe in section 2, our system can generate hierarchical verbal explanations. In other words, it can understand behavior of a target program on various grain-size. And the system holds the result of understanding as hierarchical data. Therefore we can realize a system generating visual explanations on various grain-size, by developing a method to generate a visual explanation from a result of understanding.

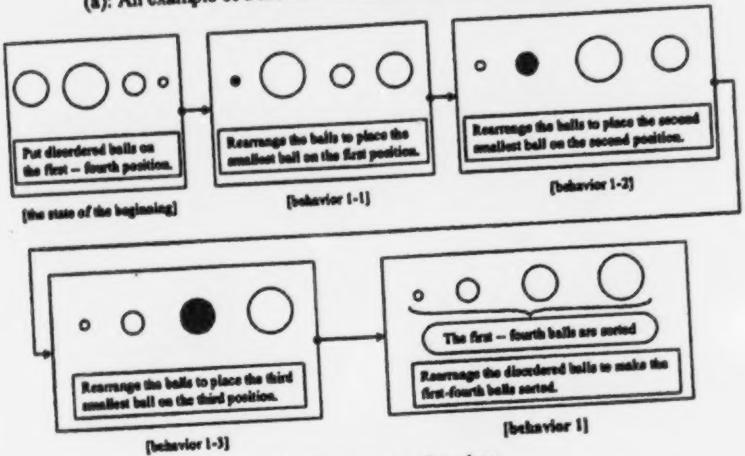
(2) The method of generating combination of verbal explanations and visual explanations.
Our program understanding mechanism can recognize the major phenomena in the domain world. And we have already developed a method to generate verbal explanations from the result of program understanding. Thus, if the system can generate a visual explanation from the result of it by the method (1), it becomes to be able to generate both visual explanations and verbal explanations remarking major phenomena from common data.

(3) The method of generating explanations on the total effect of a sequence of statements.
By generating an explanation remarking that a task is achieved just after explanations of sub-tasks are finished, the system can generate explanations on the total effect of the task. The explanations of the task and the sub-task can also be generated by the method (1) and (2). For example in Figure 4, just after the explanation corresponding to the behavior 1-3 is finished, the system generates the explanation corresponding to the behavior 1 as the explanation of the total effect. As a result, the explanation shown in Figure 4 is generated.

In consequence, if we can realize the method (1), the method (2) and (3) can also be realized. Therefore we discuss the detail of the method (1) in the next section.


Behavior 1
Rearrange the disordered balls to make the first -- fourth balls sorted
Behavior 1-1
Rearrange the balls to place the smallest ball on the first position.
Behavior 1-2
Rearrange the balls to place the second smallest ball on the second position.
Behavior 1-3
Rearrange the balls to place the third smallest ball on the third position.

(a): An example of a hierarchical structure of behaviors of a program


Put disordered balls on the first -- fourth position.
[the state of the beginning]
Rearrange the balls to place the smallest ball on the first position.
[behavior 1-1]
Rearrange the balls to place the second smallest ball on the second position.
[behavior 1-2]
Rearrange the balls to place the third smallest ball on the third position.
[behavior 1-3]
The first -- fourth balls are sorted
Rearrange the disordered balls to make the first-fourth balls sorted.
[behavior 1]

(b): Generated explanations

Figure 4: An example of a bimodal-explanation of a total effect

4.2 Generating visual explanations

The system visualizes behavior of the target program in various grain-size. The generated animations are shown with verbal explanations. The detail of our method to generate verbal explanation is seen in [6], so we omit it in this paper.

At first, the system starts explaining with the largest grain-size, then shows more detailed explanation on an action of which detail a learner wants to see.

The methods to draw a step of animation are classified into the following two types:
1) The method of visualization for a function implemented by a single statement.
2) The method of visualization for a function implemented by a sequence of statements.
The detailed process of 1) and 2) is discussed in 4.2.1 and 4.2.2 respectively.

4.2.1 How to generate a visual explanation of a function implemented by a single statement

In order to generate a visual explanation on a statement, we prepare specific procedures for each type of a statement. The statements of inputting, assignment, selection, and iteration have their individual procedures.

Procedures for inputting statement should be classified into several types in order to generate effective explanations. For example, the basic function of inputting statement "read (A);" must be "a datum is input to the variable A". However, showing only the basic function is not always a good explanation. If a meaningful datum has been stored in the variable "A" before inputting, the system should also explain that the datum is deleted by the inputting. Therefore, the procedures for inputting statement are classified according to some conditions on the role of the statement in the target program and the domain world: for example, the condition whether the datum stored in the destination variable of inputting has been referred before the input sentence or not (if it has been referred, it must be meaningful).

Similarly, procedures for assignment statement should also be classified. For example, the basic function of the statement "A:=B;" is "the datum in the variable B is copied and the copy is written on the variable A". But, if the datum in B will never be referred after the assignment, the explanation " the datum in B is moved to A" must be better because it represents the role of the statements more directly. We show an example of a condition to classify procedures for assignment statement and a procedure corresponding to the condition, by using the statements illustrated in Figure 5.

The condition and procedure for assignment statement meaning the copying process of objects in a sorting program is as follows.

Condition: [the datum in B represents an object in the world of greater and lesser]
and [B is referred during *2] and [A is also referred during *1]

Procedure: Seen in Table 1. And the visual explanation generated by the procedure is seen in Figure 6.
Table 1 also shows the templates for generating verbal explanations in this condition.

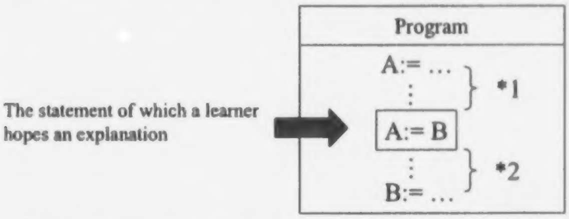


Figure 5: An example of an assignment statement in a program

The system doesn't generate visual explanations corresponding to statements of selection and iteration. For example on "if-then-else" statement, it generates visual explanations corresponding to 'then' block or 'else' block, while it generates verbal explanations whether the condition part of the statement is true or false. By the verbal explanation, a learner can understand why the 'then' block or the 'else' block is executed. Templates used to generate such verbal explanations are not the ones mentioned in 2.3 because of the following reason:

- In general, verbal explanations can be abstract. For example, we can explain a sorting process of N pieces of balls.
- Visual explanations must be concrete. For example, the system has to decide how many balls exist in the domain world in order to draw sorting process.
- Therefore, verbal explanations corresponding to visual explanations have to be generated by templates designed for bimodal-explanation.

Table 1: Examples of procedures and templates for generating explanations of an assignment statement

order	Procedures for generating visual explanations.	Templates of verbal explanations
1	Show a ball having an assigned value with a black color.	Focus on [a ball holding an assigned value].
2	Show a copy of the ball having the assigned value.	Prepare a copy of this ball.
3	Show a ball assigned the value with a gray color.	Remove a ball on [an assigned position].
4	Draw an arrow from the copy of the ball having the assigned value toward the ball assigned the value.	Move the copy of this ball to [an assigned position].
5	Show the copy of the ball having the assigned value at an assigned place.	Move this ball to [an assigned position].
6	Show a state of end.	

As mentioned above, "[]" in template means a procedure by which a certain pattern of string is generated. The bimodal-explanation corresponding to a single statement is composed of 3 ~ 5 scenes.

4.2.2 How to generate an explanation of a function implemented by a sequence of statements

In order to generate such a visual explanation, the system needs to display the states before/after the function has been applied. In order to decide designs of both states, the system generates a sequence of visual explanations on the statements by which the function is implemented. The generated explanations are only stored in a database without being displayed to a learner. The system picks up the initial state and the final state from the database and displays them one after another.

In addition, the system also needs to explain an effect of the sequence of statements directly. In order to generate an explanation of a total effect of statements, the system should generate a verbal explanation remarking the total effect and a step of animation showing the effect directly. The system can generate the verbal explanation by applying templates to the result of simulation. In order to generate a step of animation, we prepare procedures for visualizing a concept recognized as a result of the simulation. For example, brace and the words attached to it in Figure.4 (b) are drawn by such procedures. The number of such procedures is nearly equal to the number of template for the result of simulation (illustrated in 2.3.(1)).

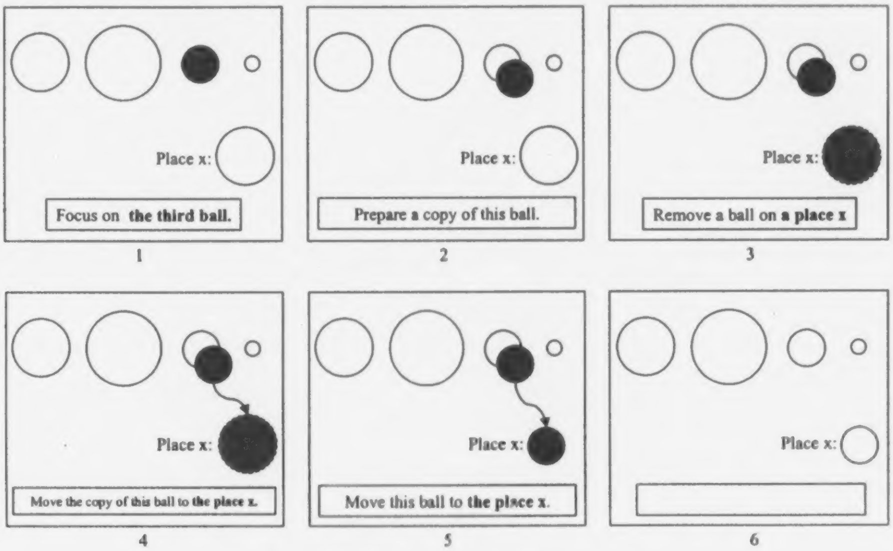

Place x:
Focus on **the third ball.**
1
Place x:
Prepare a copy of this ball.
2
Place x:
Remove a ball on **a place x**
3
Place x:
Move the copy of this ball to **the place x.**
4
Place x:
Move this ball to **the place x.**
5
Place x:
6

Figure 6: The visual explanation corresponding to each statement

5 Implementation

The system is developed on Unix workstations. The unit of generating bimodal-explanation is implemented by Tcl/Tk. Now we have finished implementation of procedures generating bimodal-explanations of statements of input, assignment, selection and iteration. Figure 7 shows an example of bimodal-explanations generated by our system. The target program is sorting by straight insertion. The system explains the process of sorting five balls according to their sizes. Figure7-1 shows the state just after that the smallest ball (in this figure, it is the 4th ball from the left end) has been found, the ball has been copied to the 'place x', and it has been removed. After that, the following processes are explained one after another.

- The copy of the first ball is moved to the fourth position (Figure7-2, 7-3).
- The ball on the 'place x' is focused (Figure7-4).
- The ball on the first place is removed (Figure7-5).
- The ball on the 'place x' is moved to the first position (Figure7-6, 7-7).

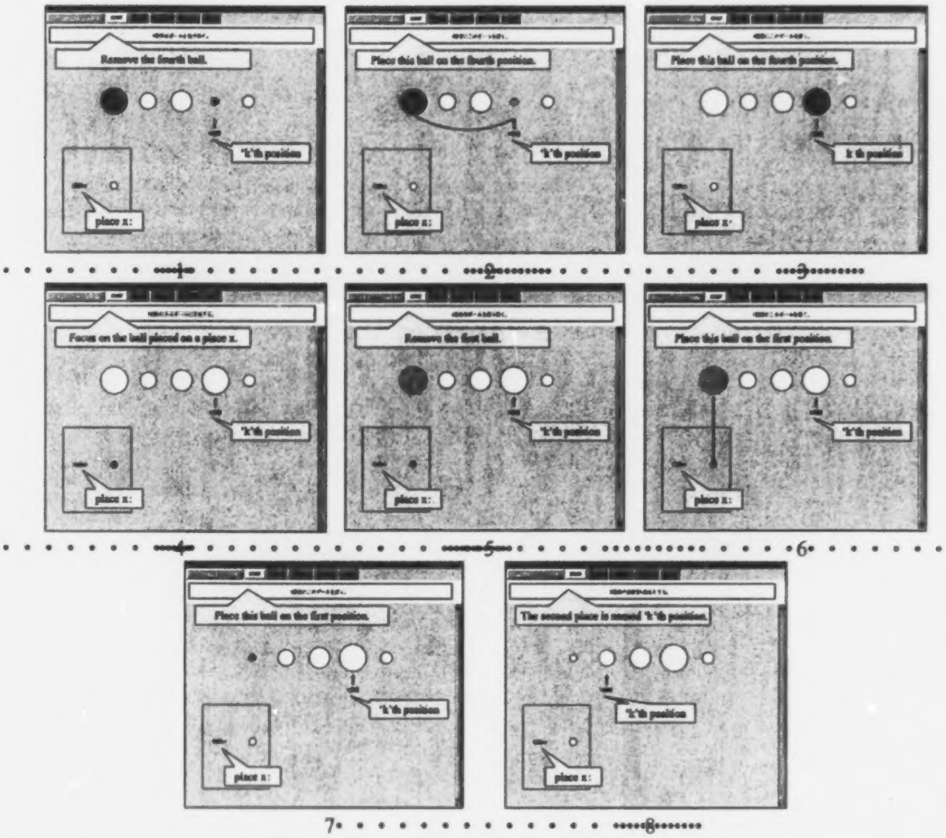
By these explanations, learners can imagine the process that the smallest ball is moved to the first place. The explanation after Figure7-8 continues in a similar way. The system also shows the process that the second - fifth smallest ball is moved to the second - fifth place respectively. Thus, the whole process of the sorting is illustrated. The original messages generated by our system are Japanese, but we add corresponding English messages to this figure.

6 Conclusion

In this paper, we proposed a method of generating a bimodal-explanation. Our system accepts the result of simulation and generates bimodal-explanations. Our current system can deal with only on the domain world of greater and lesser. Constructing procedures for the remaining types of a statement and applying this system to other domain worlds will be our future work.

Acknowledgments

This research was supported in part by Grant-in-Aid for Scientific Research (c)(11680216) from Japan Society for the Promotion of Science.



1. Remove the fourth ball.
2. Place this ball on the fourth position.
3. Place this ball on the fourth position.
4. Focus on the ball placed on a place x.
5. Remove the first ball.
6. Place this ball on the first position.
7. Place this ball on the first position.
8. The second place is named 'k'th position.

Figure 7: Outputs of our bimodal-explanation system

References

[1] Adam, A., Laurent, J.P.: "LAURA, A system to debug student programs", Artificial Intelligence, Vol.15, pp.75-122, (1980)

[2] Ando, T., Itoh, R., Konishi, T., Itoh, Y.: "Extension of the Educational System that Generates Domain-Oriented-Explanations of Programs – On Methods for Designing Domain World Models and for Simulation", Proc. of ICCE'99, Chiba, Japan, Vol.1, pp.856-860, (1999)

[3] Brown, M. H.: "Zeus: A system for algorithm animation and multi-view editing (research report No.75)", DEC systems research center, (1992).

[4] Henry, R. R., Whaley, K. M., Forstall, B.: "The university of Washington Illustrating Compiler ", Proc. of The ACM SIGPLAN'90 conference on programming language design and implementation, pp.223-233, (1990).

[5] Itoh, R., Nishizawa, M., Konishi, T., Itoh, Y.: "On analysis of novice PASCAL programs based on simulation to extract Information for education", Proc. of ICCE'97, Kuching, Sarawak, Malaysia, pp.485-492, (1997)

[6] Itoh, R., Nishizawa, M., Konishi, T., Itoh, Y.: "A programming learning system which makes Domain-Oriented-Explanations", Proc. of ICCE'98, Beijing, China, Vol.1, pp.432-440, (1998)

[7] Itoh, R., Konishi, T., Itoh, Y.: "On Constructing a programming Education System that makes domain-oriented-explanations", Journal of JSAI, Vol.15 No.2, pp.362-375, (2000), in Japanese.

[8] Johnson, W.L., Soloway, E.: "PROUST:knowledge-based program understanding", IEEE Software, vol.11, no.3, pp. 267-275, (1985)

[9] Stasko, J. T.: "Tango: A framework and system for algorithm animation", IEEE computer, Vol.23, No.9, pp27-39, (1990).

BEST COPY AVAILABLE

An Environment for Learning by Design
- In the Case of Learning of Search Algorithm -

Tsukasa HIRASHIMA, Hidenori IKEMATSU, Mayt Kalayar, Akira TAKEUCHI
Kyushu Institute of Technology, Department of Artificial Intelligence
680-4, Kawazu, Iizuka, 820-8502, JAPAN
TEL: +81-948-29-7618, FAX: +81-948-29-7601
tsukasa@ai.kyutech.ac.jp

This paper described a learning environment for search algorithms. In the learning environment, learners can build search algorithms by combining several parts by direct manipulation. Then, the environment diagnoses the algorithms in order to give feedback about the algorithms. First, the environment judges whether or not the algorithms are adequate. When the algorithms aren' t adequate, they are diagnosed using heuristics rules. In the diagnosis, errors in the algorithms are detected. By using the results of this diagnosis, the environment can give messages to help the learners revise their algorithms or to motivate them to build the next type of algorithms. We have already implemented the learning environment. As a preliminary evaluation of the environment, we asked 13 students to use the environment, and gathered several types of data. As a result, the experiment suggests that the learning environment is promising.

Keywords: **Learning by design, Error diagnosis, Search algorithm**

1 Introduction

An effective way to learn procedural knowledge in depth is to make learners apply it to various cases. However, although the learners may master how to use the procedure through the experience, it is not enough to answer the question "what the procedure is". Several investigations [1-4] suggested that "learning by design" is a promising way to promote the learner's understanding about "what that is". For example, in order to understand a machine in depth, assembling it from its smaller parts is the best way. In the case of the understanding of procedure, to build up the procedure by trial and error is useful in order to understand it.

This paper reports about a learning environment for learning by design, targeting basic search algorithms taught in an introduction to artificial intelligence lecture, that is, depth-first search, breadth-first search and three heuristics searches (best-first, minimum consuming cost, and A algorithm). In the lecture, usually, the procedure of each search algorithm is taught and learners carry out the searches following the procedures by hand. Some of them understand the meaning of the algorithms through the practice, but some of them only memorize the procedures. Our environment provides several parts of the search algorithm as icons. Learners can assemble them by direct manipulation in the environment. The environment interprets the assembled parts as a search algorithm and diagnoses it, for example, as to whether it falls into the loop or not. Then, the environment gives feedback for the algorithm to revise it or to try to build the next one. The tree structures that are generated as the results of the searches following the algorithms are also presented to the learners. These feedbacks are necessary to realize learning by design effectively.

In this paper, first, the model of the search algorithm that is the basis of the design of this environment is described. Then, the configuration and functions of the learning environment are explained. The preliminary evaluation of the environment is also reported.

2 Interactive Learning Environment of Search Algorithms

Figure 1 shows the configuration of the ILE. It is composed of the interface and reasoning module. In the interface, learners design and build search algorithms, and receive feedback from the system. In the reasoning module, the algorithms are diagnosed and feedback messages for them are generated. The interface is implemented in Java as a client and the reasoning module is implemented in Prolog as a server. Therefore, the ILE can be used on the Internet.

In this section, first, the model of search algorithms used in the ILE is described. The modeling is indispensable for designing the interface for algorithm building and in order to diagnose algorithms. Then, the interface where learners can build the search algorithms by direct manipulation is presented. The diagnosis of the algorithms and the feedback generated based on the results of the diagnosis are also explained.

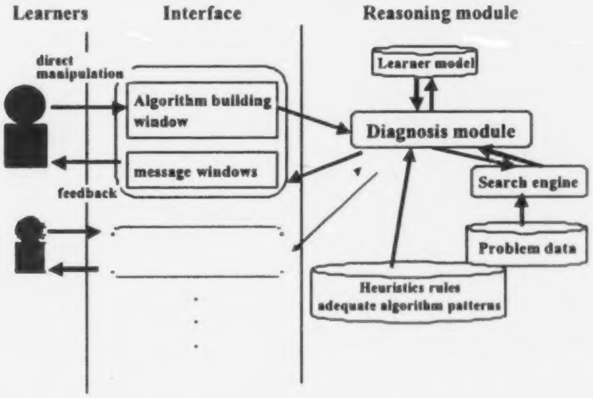


Figure 1. Configuration of the ILE

2.1 A Model of Search Algorithm

Search algorithms taught in the introductory lecture of artificial intelligence share the same procedure as follows. Here, both "Open" and "Closed" are lists composed of search nodes.
1) The start node is put into Open.
2) If "Open == []" then "the end of the search is in failure".
3) Pick up one node at the head of Open (the node is called *n*)
4) If "*n* == goal" then "the end of the search is in successful".
5) Generate child nodes from *n*.
6) Put the child nodes into Open.
7) Put *n* into Closed.
8) Return to Step-2.

The differences between the search algorithms are characterized by the operation of Step 6. For example, depth-first search is characterized as the algorithm in which the child nodes are put into the head of Open in Step 6. Breadth-first search is characterized as the algorithm in which the child nodes are put into the tail of Open in Step 6. In heuristics searches, the way to sort Open is an essential characteristic. In addition, for every algorithm, the method of selection of child nodes to put into Open is also an element that characterizes the search algorithms.

In our system, search algorithms are characterized by the combination of the following three list operations used in Step 6: "selection," "connection" and "sort." There are two types of selection operations: the first is "to select nodes that are not included in a list," and the other is "to select nodes that are not included in a list or are lower in cost than the same node in the list." Connection also has two types. The first is "to put nodes into the head of a list" and the other is "to put nodes into the tail of a list." The referred list is usually Open. We prepared three types of sorts: "to sort in the order of the consumed cost (minimum consumed cost search)," "to sort in the order of predicted cost (best-first search) " and "to sort in the order of the total of the consumed and predicted cost (A algorithm)."

BEST COPY AVAILABLE

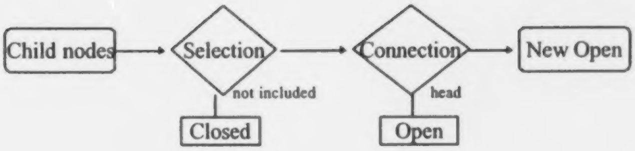

Child nodes
Selection
not included
Closed
Connection
head
Open
New Open

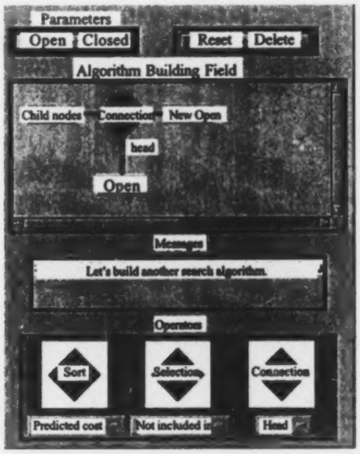
Figure 2. An example of the model of Search Algorithm

Figure 2 shows an example of a search algorithm built by the operations. The lozenge is the operation, and the rectangle is the list. The parameter that indicates "referred cost" or "head or tail" to specify the operator is presented at the bottom right of the lozenge. Therefore, Figure 2 means that "the child nodes that are not included in Closed are put into the head of Open." This is a kind of depth-first search that prunes using Closed.

Every part described above is necessary to build the search algorithms taught in the introductory lecture to artificial intelligence. In order to make learners understand search algorithms more deeply, our ILE provides an environment where learners can build search algorithms freely, and can receive feedback for the algorithms. In the following section, the ILE designed based on the model of search algorithms is described.

2.2 Building Search Algorithms

The interface for building search algorithms is shown in Figure 3 (currently, the interface is written in Japanese. Explanations in Figure 3 are translated to English for this paper. Japanese version is shown in [5]). Learners build search algorithms in the "building field" by assembling parts provided in the interface. At the bottom of the Interface, three operators are provided in the lozenges. The parameters specifying the operators are selected from the menu under the lozenges. The reference lists of the operators are selected from the box at the upper left. All manipulation in the interface can be done with a mouse. The algorithm in the building field is a depth-first search without having pruned.


Parameters
Open
Closed
Reset
Delete
Algorithm Building Field
Child nodes
Connection
New Open
head
Open
Messages
Operators
Sort
Selection
Connection
Predicted cost
Not included in
Head

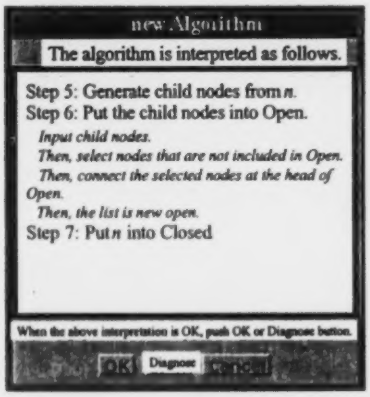
Let's build another search algorithm.

Figure 3. The Algorithm Building Field.

Learners can confirm the algorithm built by themselves in two ways: a written explanation and a trace of the

BEST COPY AVAILABLE

search tree. The explanation is generated by interpreting the operations in order of sequence in the building field. Figure 4 is the explanation of the algorithm shown in Figure 3. A search tree is generated by showing the trace results in a search space. The search spaces are provided as mazes in the environment. Figure 5 is an example of search tree that is the results of the search for the maze shown in the right in the figure.



The algorithm is interpreted as follows.

Step 5: Generate child nodes from *n*.
Step 6: Put the child nodes into Open.
Input child nodes.
Then, select nodes that are not included in Open.
Then, connect the selected nodes at the head of Open.
Then, the list is new open.
Step 7: Put *n* into Closed

When the above interpretation is OK, push OK or Diagnose button.

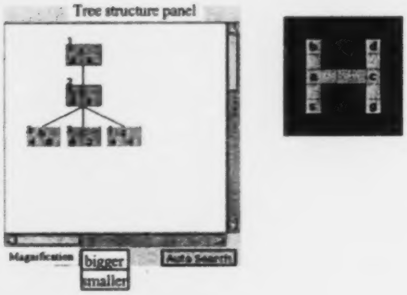
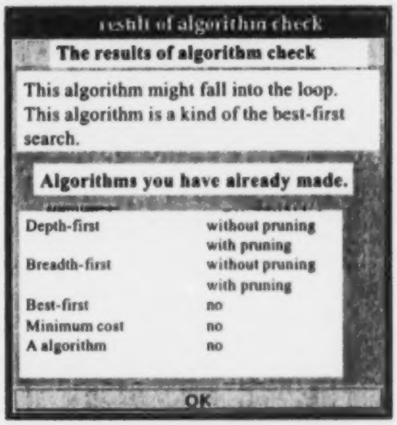


Figure 4. An Example of Explanation an Algorithm. Figure 5. An example of search tree.



The results of algorithm check

This algorithm might fall into the loop.
This algorithm is a kind of the best-first search.

Algorithms you have already made.

Depth-first	without pruning with pruning
Breadth-first	without pruning with pruning
Best-first	no
Minimum cost	no
A algorithm	no

Figure 6. An Example of the Results of Problem Diagnosis.

Learners can also ask the system to diagnose the algorithms built in the building field. The reasoning module has both the adequate combinations of operations and heuristics rules to criticize the algorithms that are not adequate. By using the adequate combinations, the adequate algorithms can be detected. By using the heuristics rules, the errors in the inadequate algorithms are detected. If no errors are detected by the heuristics rules, the reasoning modules can not judge the type of the errors. The heuristics rules are prepared from the following three points of view: the kind of algorithm, redundancies in the algorithm and the covering of the search space. An example of messages generated from the results of the diagnosis is shown in Figure 6. In the following section, the diagnosis of the search algorithms is described.

2.3 Diagnosis of Search Algorithms

In the reasoning module, the algorithms are diagnosed using heuristics rules. The heuristics rules of each viewpoint are shown in this section.

2.3.1 Type of Algorithm

The algorithms built by the learners are categorized by the following heuristics rules.

- When, after child nodes are put into the head of Open, either any nodes are not put into the head of Open or Open is not sorted, the algorithm is categorized as depth-first search.
- When, after child nodes are put into the tail of Open, either any nodes are not put into the tail of Open or Open is not sorted, the algorithm is categorized as breadth-first search.
- When, after child nodes are put into Open, Open is finally sorted in the order of consumed cost, the algorithm is categorized as minimum consumed cost search.
- When, after child nodes are put into Open, Open is finally sorted in the order of predicted cost, the algorithm is categorized as predicted cost search.
- When, after child nodes are put into Open, Open is finally sorted in the order of the total of consumed cost and predicted cost, the algorithm is categorized as A algorithm.

When the algorithm has no characteristics checked by the above rules, the kind of algorithm cannot be specified.

2.3.2 Redundancy of Algorithm

When the algorithms include the following operators, the diagnosis module judges that the operators are redundant in the algorithms.

- The same operators are used continuously.
- When several operators of sort are used, only the operator of sort used at the end has meaning.
- After using the connecting operator with a list as the parameter, the execution of the selection operation with the same list as the parameter results in deleting the added nodes.

2.3.3 Covering of the Search Space

Several search algorithms that can be built by learners can not find goals that exist in a search space. The reasoning module diagnoses whether or not the algorithm can cover the search space, by using the following heuristics rules.

- When several child nodes which might imply goals are not put into Open, the algorithm might fail to reach the goal included in the search space.
- When the algorithm that isn't categorized as breadth-first, minimum consuming cost or A algorithm doesn't include the selection operator with Closed as the parameter, the algorithm falls into the loop.

2.4 Feedback based on the Diagnosis

Based on the results of the above diagnosis with heuristics rules, the messages to criticize the algorithm are provided in the interface. Figure 7 shows an example of the messages. When the type of an algorithm is judged, the type is indicated. When the algorithm includes the redundant operators, the operators and the explanation of the redundancies depending on each heuristics rule are provided. When the algorithm might not cover the search space because several child nodes fail to be input into Open, the explanation prepared for the heuristics rule is shown. When the algorithm might fall into the loop, the possibility of falling into the loop is indicated.

When the algorithm includes a pruning operation, the fact is also indicated. In the interface shown in Figure 7, to motivate learners to build the next algorithms, the algorithms the learner has made correctly and hasn't made yet are shown.

3 Preliminary Evaluation

For a preliminary evaluation of the learning environment, we gathered thirty college students and asked them to use the learning environment. Those who were in the second grade or in the third grade have already taken the lecture of artificial intelligence. Their participation was voluntary. Before the experiment, we explained how to operate the environment for ten minutes. Then, we asked them to build search algorithms in the learning environment for an hour.

In the experiment, we recorded the following data: (1) the number of algorithms built by the learners, (2) the number of adequate algorithms, (3) the number of inadequate algorithms that could be diagnosed with heuristics rules, (4) the number of inadequate algorithms that couldn't be diagnosed, and (5) the number of types of the adequate algorithms the learner made. The results are shown in Table 1. After the experiment, we asked four questions: (a) Are you interested in the system? (b) Is the system easy for you to use? (c) Would you like to use the system more? (d) Do you understand the search algorithms better than before? The results are shown in Table 2.

Table 1. The results of the students algorithm building.

Student number	(1)The total number of algorithms	(2)Adequate algorithms	(3)Inadequate algorithms (be diagnosed)	(4)Inadequate algorithms (not be diagnosed)	(5)the type of the algorithms
No.1	26	11	15	0	5
No.2	24	10	6	8	5
No.3	16	10	6	0	5
No.4	38	17	13	8	5
No.5	44	13	20	11	5
No.6	48	10	26	12	5
No.7	21	8	8	5	3
No.8	26	15	9	2	5
No.9	23	14	4	5	5
No.10	16	5	6	5	1
No.11	43	24	11	8	5
No.12	20	8	12	0	5
No.13	16	9	7	0	5
	361	154	143	64	

Table 2 . The Results of Questions.

	Yes	Maybe yes	No	No answer
Qusetion-a	10	3	0	0
Question-b	1	7	5	0
Question-c	7	5	0	1
Question-d	3	3	4	3

In Table 1, the total number of algorithms the learners made was 361, that is, 27.8 per student. The total number of adequate algorithms was 154, that is, 43 % of the algorithms. The total number of inadequate algorithms was 207 (57%). The number of diagnosed errors by heuristics rules was 143. This means that the system could detect the errors in 69 % of the inadequate algorithms. Among thirteen students, eleven students made every type of algorithm.

In Table 2, the results of Question-a and –c suggest that most of the students had interest in the learning environment. The result of Question-b indicates that the interface is not easy for the students to use. For Question-d, four students answered "no", and three students didn't judge, that is, more than half the students didn't think they gained a deeper understanding by using the learning environment.

Students made many algorithms in the experiment and they answered that the learning environment was interesting. In addition most of them could make every type of algorithm. These results suggest that the learning environment is promising. The answers for Question-b mean we should improve the interface. In Question-d, Six students thought they got deeper understanding by using the environment, but seven students didn't think so. When we gathered students, we told them that we would ask them to use a learning environment for search algorithms. Therefore, most of the students participating in the experiment might have confidence about their understanding of search algorithms. This is one reason for the result for Question-4.

2.3.1 Type of Algorithm

The algorithms built by the learners are categorized by the following heuristics rules.

- When, after child nodes are put into the head of Open, either any nodes are not put into the head of Open or Open is not sorted, the algorithm is categorized as depth-first search.
- When, after child nodes are put into the tail of Open, either any nodes are not put into the tail of Open or Open is not sorted, the algorithm is categorized as breadth-first search.
- When, after child nodes are put into Open, Open is finally sorted in the order of consumed cost, the algorithm is categorized as minimum consumed cost search.
- When, after child nodes are put into Open, Open is finally sorted in the order of predicted cost, the algorithm is categorized as predicted cost search.
- When, after child nodes are put into Open, Open is finally sorted in the order of the total of consumed cost and predicted cost, the algorithm is categorized as A algorithm.

When the algorithm has no characteristics checked by the above rules, the kind of algorithm cannot be specified.

2.3.2 Redundancy of Algorithm

When the algorithms include the following operators, the diagnosis module judges that the operators are redundant in the algorithms.

- The same operators are used continuously.
- When several operators of sort are used, only the operator of sort used at the end has meaning.
- After using the connecting operator with a list as the parameter, the execution of the selection operation with the same list as the parameter results in deleting the added nodes.

2.3.3 Covering of the Search Space

Several search algorithms that can be built by learners can not find goals that exist in a search space. The reasoning module diagnoses whether or not the algorithm can cover the search space, by using the following heuristics rules.

- When several child nodes which might imply goals are not put into Open, the algorithm might fail to reach the goal included in the search space.
- When the algorithm that isn't categorized as breadth-first, minimum consuming cost or A algorithm doesn't include the selection operator with Closed as the parameter, the algorithm falls into the loop.

2.4 Feedback based on the Diagnosis

Based on the results of the above diagnosis with heuristics rules, the messages to criticize the algorithm are provided in the interface. Figure 7 shows an example of the messages. When the type of an algorithm is judged, the type is indicated. When the algorithm includes the redundant operators, the operators and the explanation of the redundancies depending on each heuristics rule are provided. When the algorithm might not cover the search space because several child nodes fail to be input into Open, the explanation prepared for the heuristics rule is shown. When the algorithm might fall into the loop, the possibility of falling into the loop is indicated.

When the algorithm includes a pruning operation, the fact is also indicated. In the interface shown in Figure 7, to motivate learners to build the next algorithms, the algorithms the learner has made correctly and hasn't made yet are shown.

3 Preliminary Evaluation

For a preliminary evaluation of the learning environment, we gathered thirty college students and asked them to use the learning environment. Those who were in the second grade or in the third grade have already taken the lecture of artificial intelligence. Their participation was voluntary. Before the experiment, we explained how to operate the environment for ten minutes. Then, we asked them to build search algorithms in the learning environment for an hour.

As for the results, the experiment suggests that the learning environment is promising to be used in the real world, but the effect couldn' t be confirmed clearly.

4 Conclusions

This paper described a learning environment for learning by design in the case of search algorithms. In the learning environment, learners can build search algorithms by combining parts by direct manipulation. Then, the environment diagnoses the algorithms in order to give feedback about the algorithms. First, the environment judges whether or not the algorithms are adequate. When the algorithms aren' t adequate, they are diagnosed using heuristics rules. The heuristics rules detect errors in the algorithms. By using the results of this diagnoses, the environment can give messages to help the learners revise their algorithms or to motivate them to build the next type of algorithms.

We have already implemented the learning environment. As a preliminary evaluation of the environment, we asked 13 students to use the environment, and gathered several types of data. As a result, the experiment suggests that the learning environment is promising to be used in the real world and that is promising, but the effect couldn' t be confirmed clearly. In the next step, we will use the learning environment in class and evaluate it in a real learning context.

References

(1) S.Papert: Mindstorms, Basic Books, Inc(1980).

(2) G.Fisher, K.Nakakoji: Empowering Designers with Integrated Design Environments, Proc. of the First International Conference on Artificial Intelligence in Design, pp105-117(1991).

(3) A.Repenning: Programming Substrates to Create Interactive Learning Environments, Interactive Learning Environments, Vol.4, No.1, pp.45-74(1994).

(4) E.Soloway, M.Guzdial, K.Hay: Learner-Centered Design: The Challenge for HCI In The 21st Century Interactions,Vol.1, No.2, pp.36-48(1994).

(5) H.Ikematsu, Myat, T.Hirashima, A.Takeuchi: Interactive Learning Environment for Search Algorithm, Proc. of Research Meeting of Kyushu Division of Information Processing Society of Japan, 3A-5, pp.101-108(2000)(in Japanese).

An on-line ITS for elementary algebra

Li Yi-Ping*, Chan Kan-Kan**

University of Macau
P.O.Box 3001, Macau
Tel: 853-3974323
Fax: 861095
*fstypl@umac.mo , **fedrc@umac.mo

The objective of this research is to reinforce the concepts and procedures of elementary algebra that students learn in junior schools. Students react to teacher's instructions in various way. However, in a traditional class, the ratio of teacher to student is still too great. The question of how to help all the students with limited number of teachers arises. This paper describes how to achieve the above objective with the help of an intelligent tutoring system. It discusses the design outline and the system architecture of the proposed system. The tutor tracks student's performance and uses this information to provide most suitable instruction to each student dynamically.

Keywords: **Web-based learning environment, intelligent tutoring system, elementary algebra**

1. Introduction

In a class of forty students, it is hard for teachers serving every student's questions within a class of forty minutes. Teachers teach students concepts and methods or techniques to solve problems in group. Then related exercises are given to students to practice at home. Students who have no doubts in class might cope with the exercise and learn well while some might not master the technique that teacher has taught. They always frustrate when they cannot solve the problem. In this situation, some advice from teacher is very helpful in their learning process. However, teachers are not always available while they need help. Also teachers might not be able to answer many students' doubts at the same time. This research is conducted with the aim of using computers to support the knowledge acquisition process that is adjusted to the capabilities of individual student. Students just need to have a web browser to connect to school network and would get assistance right away.

Many existing CAI applications do help a bit in students' learning process but they do not consider the background knowledge of students. This means that they might provide inappropriate feedback to students which in turn affects student's progress in learning. In order to overcome this situation, research has been investigated on intelligent tutoring system which includes functions for guiding students towards proper knowledge acquisition, according to observation of the student's problem-solving process and identification of the causes of student error.

We first depict the learning environment of our system in section 2and then the overall architecture of our system is mentioned with detail description of main components of ITS in section 3. The final section concludes our work.

2. Learning Environment of on-line ITS for elementary algebra

ITS for elementary algebra is designed as a problem solving environment to be used in class. Therefore we assume that student is familiar with the basic concepts of elementary algebra and know the ways to factorize a polynomial. Students use the system as a tool at home or during class practice. Since the condition that students use it lacks teacher's support, an interactive problem support should be built into the system. With this feature, students might get help on steps of problem solving where he has difficulty.

In order to access the on-line tutoring system, a student just needs a web browser and types the address where the system locates. An instance of the system will be created in student's computer in the form of ActiveX control. Although it might argue that there is great network delay in loading the system in student's computer, interactivity and userfrindliness deserve a short delay. In fact, in a local environment, the network traffic is not so congested. Therefore this is not a real problem. Instead, students can use it as if any Windows program and do not have to worry about its maintenance or compatibility issues. The control serves as a communicator betweent the system and the student. It transfers student's action to the system and returns the reponse of the tutor to the student.

Every student has his own session during the learning process. When a student enters the system with his user name and ID, a model of student performance is created or opened to set his learning environment. ITS selects a problem according to student's level for him to work or waits for the student to enter a problem which he has doubts. In both situations, the student solves it with the guidance of the tutor in a step-by-step way. The system keeps track of every step of the student in background. If nothing goes wrong, it remains quiet otherwise it prompts student's error. His problem solving procedure is kept in the system for future reference.

3. Overview of on-line ITS for elementary algebra

Our system follows the standard architecture of client-server model. The system resides on the server side. The basic components of the learning system are the domain module, pedagogical module, student modeler and the interface.

The domain expert module consists of two main programs. One is a problem solver which is capable of solving problems in its knowledge base. The other one is question generator that creates new problems according to the instruction of pedagogical module. In order to achieve its mission, the knowledge base is composed of both rules and cases. The expert model is capable of solving general problems by the rules coded in its module. As for miscellaneous problems, they are indexed as cases with problem characteristics and solving techniques so that the domain expert knows how to retrieve the relevant solving technique with the detected problem features.

The domain that we have chosen is factorization of algebraic polynomial for students in elementary classes. Given a polynomial, factorization is to express an integral polynomial as a product of prime polynomial. Therefore a polynomial is not completely factored unless each factor is either a monomial or a prime integral polynomial. Generally, there are 4 basic methods to factorize an algebraic formula. They are: (1) obtaining the common factor (2) using identities (3) cross-method and (4) divide the polynomial into groups and then simplify groups to find factors.

The pedagogical goal is to let junior students master the methods to factorize a polynomial smoothly. Students are taught the basic method to factorize an algebraic formula. However, they always get lost in the actual application to find the factors of a given formula. Therefore, we have organized the pedagogical knowledge by constructing groups of problems according to the level of difficulty, problem characteristics and solving technique. Within each level, there are pre-requisite question types which a student must understand before a certain question type will be generated. Figure 1 shows part of the relation among question types. There are several groups having polynomial problems as bellows:

- Problems, which just need one method to solve. They are polynomial with common factors, problems that satisfies the characteristics of perfect square: $(a\pm b)^2 = a^2\pm 2ab+b^2$, difference of 2 squares: $a^2-b^2=(a+b)(a-b)$, sum or difference of 2 cubes: $a^3\pm b^3 = (a\pm b)(a^2\mp ab+b^2)$, or perfect cube: $(a\pm b)^3 = a^3\pm 3a^2b+3ab^2\pm b^3$ and problems of trinomials with a degree of 2 i.e. $x^2+(a+b)x+ab = (x+a)(x+b)$
- Problems that need 2 methods to solve are posed, for example: ab^2-4a. There are a few combination of solving techniques like common factor with standard equation, common factor with cross method or cross method with standard equation.
- Problems with more than 4 terms that need to be divided into groups of terms before they can be solved by the general methods.
- Problems that require special techniques to solve like adding terms, splitting terms etc.

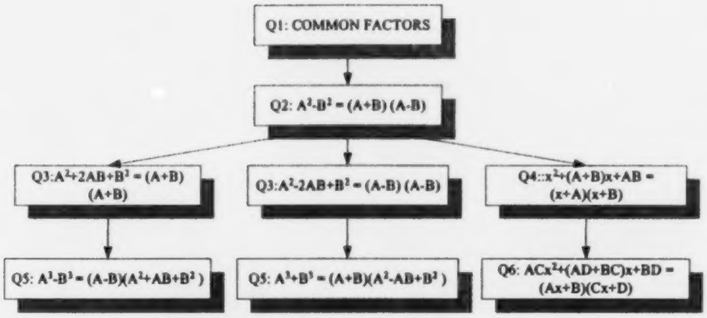

Q1: COMMON FACTORS
Q2: $A^2-B^2 = (A+B)(A-B)$
Q3: $A^2+2AB+B^2 = (A+B)(A+B)$
Q3: $A^2-2AB+B^2 = (A-B)(A-B)$
Q4:: $x^2+(A+B)x+AB = (x+A)(x+B)$
Q5: $A^3-B^3 = (A-B)(A^2+AB+B^2)$
Q5: $A^3+B^3 = (A+B)(A^2-AB+B^2)$
Q6: $ACx^2+(AD+BC)x+BD = (Ax+B)(Cx+D)$

Figure 1. Part of the pedagogical knowledge

A student modeler tries to understand the mental state of a student so as to provide a more accurate estimation of individualized instruction. The task of building a student model is extremely difficult as the amount of information to capture is huge. Although it has been pointed out that this task is intractable[1], an incomplete student model is still very useful in the process of tutoring [2] [3].

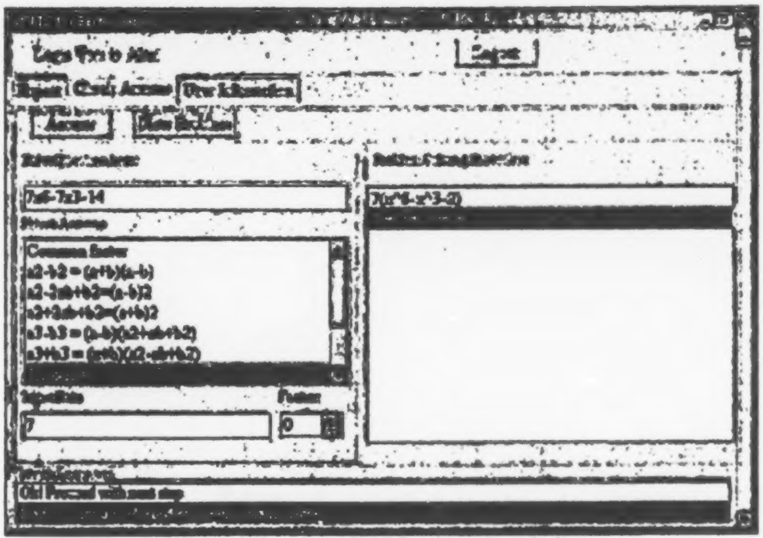
The student modeler evaluates the solution of the student and the ways he factorizes the polynomial with respect to the one solved by domain expert. Although the solution path for a given problem of a student might be different from that of domain expert, the student's solution is still correct if it answers to the problem. In our case, if all the factors that the student found are irreducible, his answer is correct. The tutor would suggest him another way to solve the problem if it is found that his solution path is different. In this way, students are guided to know that there is always another way or a better method to solve a problem. Referred to table 1, student is asked to factorize a problem $4a^2-16b^2$, the second column shows the ways that he solves the problem. The student answers the question correctly and his student model is updated accordingly. Although his problem solving procedure differs from the sample, this would not affect his student model. Only the tutor would suggest him its way for the student as reference.

Problem: $4a^2-16b^2$			
Student	Reason	Tutor	Reason
$=(2a-4b)(2a+4b)$	$a^2-b^2=(a+b)(a-b)$	$=4(a^2-4b^2)$	Common factor
$=2(a-2b)2(a+2b)$	Common factor	$=4(a+2b)(a-2b)$	$a^2-b^2=(a+b)(a-b)$
$=4(a-2b)(a+2b)$	Multiply numbers		

Table 1. Procedure that the student and the tutor solves a problem.

The student model in ITS for elementary algebra contains a general information about the student, history of student's performance such as previously solved problem, information about the usage of factorization techniques and what kind of problems he is able to solve according to the pedagogical knowledge. All these information is important to allow students to receive more instruction and perform more problem-solving questions in areas in which they are relatively weak. An array of integer is used to keep the system's belief of student's mastery of a certain skill.

The interface of our system shown in Figure 2 is designed to be user friendly. It is divided into 3 main regions: upper part shows "Check answer" and "New problem" buttons; lower part is the area where the tutor provides feedback. The student interacts with the system mainly at the left side of middle part of interface. He may enter a question by himself or the system might generate one based on his experience. A list of actions is listed for him to explore the problem solving technique. He may select an action to tell the system how he would solve the problem. Every action selected would be given an appropriate feedback to the student. In this way, he might discover what is the consequence of selecting an action. An input area is allocated for the student to enter auxiliary data needed for his selected action. When the answer button is clicked, the student's solution is evaluated and his student model is updated accordingly.


7x6-7x3-14

Common factor
a2-b2 = (a+b)(a-b)
a2-2ab+b2=(a-b)2
a2+2ab+b2=(a+b)2
a3-b3 = (a-b)(a2+ab+b2)
a3+b3 = (a+b)(a2-ab+b2)

7

0

7(x^6-x^3-2)

Ok! Proceed with next step

Figure 2. Interface of ITS for elementary algebra

3.1 Evaluation

In our experiments to simulate the problem solving procedure of students using the system, we found that it follows the overall design. It is able to provide individualized instruction, appropriate feedback and model student's performance. For major types of the factorization problem in junior school, the tutor is able to solve and guide the students. However, there are also questions that it fails to solve and guide. There are also cases that the available action for students to use in the problem solving process is not enough.

4. Conclusion

In this paper we have described an on-line intelligent system with interactive problem solving support and curriculum sequencing. A prototype system designed with some learning theory is implemented. The system helps students to reinforce the factorization technique. Our intention of building this system is to increase the learning progress of students and it shows to be a successful tool according to informal evaluation

Since the success of an ITS depends greatly on the student model, we are planning to improve our system with a more accurate student model in the near future. The user interface will be reconstructed to improve the interactivity between users and the system. The implemented domain knowledge is quite limited in this stage and we are developing larger domain knowledge.

Reference

[1] Self J., "Bypassing the intractable problem of student modeling", Intelligent tutoring system: at the Crossroads of Artificial Intelligence and Education, (1990).

[2] Self J., "Formal Approaches to Student Modeling", Student Modeling: The key to individualized knowledge-based instruction, (1994).

[3] Woolf B.P. and Murray T., "Using Machine Learning to advise a student model", Student Modeling: The key to individualized knowledge-based instruction, (1994).

BEST COPY AVAILABLE

Automatic Background Knowledge ConstructionUsing Genetic Algorithms

Jeffrey T. Chua and Raymund C. Sison
E-mail: ccsjc@ccs.dlsu.edu.ph; ccsrcs@ccs.dlsu.edu.ph
College of Computer Studies, De La Salle University, Professional Schools, Inc.
Manila, Philippines

This paper is a work-in-progress report on a proposed method for automating the task of background knowledge construction (including its possible extension) using an unsupervised model of learning. More specifically, we propose to apply the genetic algorithm operations of crossover and mutation to discover the discrepancies (bugs) between a correct solution and a set of student solutions. To illustrate the approach, we consider as a very simple example binary subtraction as problem domain.

Keywords: **buggy, crossover, mutation, GA**

1 INTRODUCTION

An Intelligent Tutoring System (ITS) is a computer program for educational support that can diagnose problems of individual learners. This diagnostic capability enables it to adapt instruction or remediation to the needs of individuals [6].

The background knowledge of an ITS is composed of the correct model of the solution and a collection of misconceptions – which is referred to as bug library. This background knowledge is constructed and is dependent specifically on the problem domain in consideration. There are problem domains wherein the background knowledge (more specifically, the misconceptions) can be completely specified, for example, arithmetic operations. For more complex problems, however, a complete specification is not a trivial activity, and in fact may not be achievable at all. In such cases, the best alternative is to enable the system to have the capability to extend its background knowledge.

The construction of the background knowledge starting from zero, and its further extension are difficult tasks. Considering that ITS is a field of computer science, it is ironic to note that the common practice is to perform the above-mentioned tasks by hand. Development of tools or methodologies that will help if not completely automate the construction and extension of the background knowledge is an important research problem.

Currently, there are very few systems with such capabilities; these include PIXIE [7], ASSERT [1], and more recently MEDD [6]. These systems use Artificial Intelligence (AI), more specifically, machine learning techniques, in the automated construction of the background knowledge.

Genetic algorithm [4] is model of learning using evolution as inspired by nature as the basis of their design and implementation. So far, it has not yet been (fully) studied whether an automated background knowledge construction can be achieved using unsupervised learning via genetic algorithms. In this paper, we propose a method for automatic background knowledge construction using genetic algorithms.

The paper is organized as follows. A brief discussion of genetic algorithms is given in section 2. It is then followed in section 3, by a description of the general framework of the proposed automatic background knowledge construction scheme. Section 4 presents a simple example considering binary subtraction as the problem domain to partly illustrate the proposed approach. Finally, the concluding remarks are given in Section 5.

2 GENETIC ALGORITHMS

Genetic algorithm is a computational model of machine learning inspired by evolution in nature. The computer based model is realized by describing a population of individuals represented by chromosomes. A chromosome is in turn (simply) described as a string that models the genetic information (also known as DNA). The species in the population are subjected to genetic operations such that after several generations the string pattern would have changed, thus modeling the concept of evolution.

There are basically two genetic operations, namely crossover and mutation. The crossover (also called recombination) operation computes the genetic information of an offspring as the crossover combination of the chromosomes of its parents. The main reason behind crossover is that, genetic information of "fit" parents should be passed on to their offspring. On the other hand, mutation occurs when a portion of the genetic information is altered (not due to crossover), sometimes in random. Mutation is the operation that allows the population of individuals to become more diversified. It should be noted that genetic information about individuals are tested against some fitness function. If the individual is tested to be "fit" then its genetic information (or part of it) are passed onward to the next generation.

3 PROPOSED METHOD FOR AUTOMATIC BACKGROUND KNOWLEDGE CONSTRUCTION

We propose the application of genetic algorithms to the problem of automatic background knowledge construction and extension. The general framework of our approach is illustrated in Figure 1.

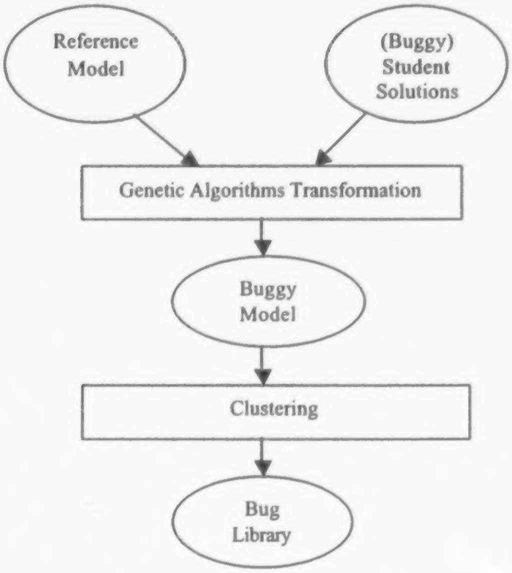


Figure 1. General Framework of Our Approach

The inputs are (i) the correct model of the problem domain, referred to as the *reference model* and (ii) the student's (possibly buggy) solutions to the problem. A model can be viewed as a set of rules for solving problems in a domain. In our proposed approach, a sequence of genetic algorithm operations (i.e., crossover and mutation) are applied on the reference model to compute a buggy model that covers all or most of the student's buggy solutions. A buggy model may possibly contain multiple bugs. Thereafter, the bugs in the buggy model are subjected to a clustering algorithm such as COBWEB [3] or MMD [6] to determine the primitive bugs. The output is a set of primitive and compound bugs to be placed in a bug library.

Encoding

A series of bits will be used to represent the individual rules that make up a model for a particular domain Notice that a rule will probably need to be tested for multiple conditions joined by logical AND operator. The rule will have to be broken down to several sub-rules. Each sub-rule will test only for an atomic condition.

The (sub)rules that were followed to perform a correct subtraction is encoded as a string. Another string that encodes the (possibly buggy) subtraction solution of a student will also need to be found. It is possible that the student' s solution may have violated more than one rule. A collection of such "buggy" strings will comprise a discrepancy set.

GA Transformations

The initial population is set to the reference model which represents the rules for a domain. The buggy student solutions are then examined to determine which rules in the reference model were violated. Frequency count are computed per rule violated. GA operations are then performed on the most frequently violated rules, and the fitness function, which is equal to the percentage of the buggy solutions satisfied is computed. Then evaluate the fitness function again. If the fitness function satisfies a user-redefinable parameter, the procedure terminates; else GA operations are performed again.

Step 1. Buggy Model reference model

Step 2. Determine frequency of violated rules

Step 3. Perform GA operations (crossover or mutation) on frequently violated rules in buggy model

Step 4. Compute for fitness function

Step 5. Evaluate fitness value of buggy model

Step 6. If fitness function > user-redefinable parameter, then stop; and return the buggy model; else go back to Step 3.

Clustering

It is possible that multiple bugs are encoded in a single model. Moreover, there are strings that would contain possibly the same set of bugs. To group similar bugs, we propose to use a clustering scheme, for example COBWEB or MMD, to separate primitive bugs from compound bugs and results are stored in the bug library.

4 ILLUSTRATIVE

To illustrate our approach, we consider as example, a simple problem of binary subtraction as the target problem domain. Binary subtraction may be performed by using "borrows" just as in the decimal system. In binary subtraction, a borrow is made from 10; thus, 10 -1 = 1. For example,

```
  10  10      - borrows

 1 0   0 1    - minuend
 0 1   1 0    - subtrahend
 -----------
 0 0   1 1    - difference
```

Binary subtraction can be summarized using the following table:

Rule	Minuend	Subtrahend	Previous Borrow	Borrow	Difference
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	0	1	1
4	0	1	1	1	0

5	1	0	0	0	1
6	1	0	1	0	0
7	1	1	0	0	0
8	1	1	1	1	1

The table shown above represents the reference model, which serves as the initial buggy model.

Given the buggy student solutions shown below,

1] 110
- 011

100

2] 101
-011

100

3] 100
-011

100

Note that the first rule that was violated was rule number 3 in the table above. Performing a GA operation, specifically mutation, we would now have a new buggy rule 3 which is:

Rule	Minuend	Subtrahend	Previous Borrow	Borrow	Difference
3'	0	1	0	1	0

One we now computes for the fitness function given the revised buggy model as equal to the percentage of buggy solutions satisfied. If the fitness value is greater than a user-redefinable parameter then we stop; else we perform GA operations again as above.

An approach to automatic bug library construction and extension using genetic algorithms was outlined, and illustrated using the problem of binary subtraction. Much has still to be done, but we think that our preliminary work on the problem presented is worth pursuing.

REFERENCES

[1] Baffes, P. and Mooney, R. "Refinement-based Student Modeling and Automated Bug Library Construction", Journal of Artificial Intelligence in Education, 7(1), 75 – 116, (1996).

[2] Burton, R. "Diagnosing Bugs in a Simple Procedural Skill", In D. Sleeman and L. Brown (Eds.), Intelligent Tutoring Systems. London: Academic Press, (1982).

[3] Fisher, Douglas, "Knowledge Acquisition Via Incremental Conceptual Clustering", Machine Learning, 2, 139 – 172, (1987).

[4] Falkenauer, Emmanuel, "Genetic Algorithms and grouping problems", John Wiley, New York, (1997).

[5] Rendell, L. "Genetic Plans and the Probabilistic Learning System: Synthesis and Results", Proceedings of the First International Conference on Genetic Algorithms and their Applications, (1985).

[6] Sison, R., Numao, M. & Shimura, M. "Multistrategy Discovery and Detection of Novice Programmer Errors", Machine Learning, 38, 157-180, (2000).

[7] Sleeman, D., Hirsh, H., Ellery, I., and Kim, I. "Extending Domain Theories: Two Case Studies in Student Modeling", Machine Learning, 5, 11-37, (1990).

AWETS: An Automatic Web-Based English Testing System

Zhao-Ming Gao
Department of Foreign Languages and Literatures
National Taiwan University
Taipei 106, TAIWAN
E-mail: zmgao@ccms.ntu.edu.tw

Test items are traditionally created by experts. While this approach has many advantages, it is laborious and time-consuming. Recent advance in corpus-based computational linguistics has shed new light on the feasibility of a computer-based language testing system capable of automatically generating items. This paper describes AWETS, an automatic web-based English testing system developed by the author's research team and used in his freshman English classes at National Taiwan University. AWETS automates test item generation, test delivery, scoring, and record keeping. It can generate random items for each testee in accordance with the input conditions of the test administrator. With AWETS, testers' jobs are reduced to inputting information such as a list of words and the time limit of each question. Besides being a useful tool for creating achievement tests in English vocabulary, AWETS can also generate proficiency tests based on a selected difficulty level without the need to input a word list. AWETS can be seen as a significant step toward future computer-based language testing system.

Keywords: **automatic generation of items, computer-based language testing, corpus-based computational linguistics, vocabulary testing**

1 Introduction

Test databank in current computer-based language testing systems is mostly created by human experts. This procedure is laborious and time-consuming. Moreover, since test databank is difficult to adapt, teachers using the systems have to spend a lot of time creating the tests for their own classes. To solve this problem, several researchers have suggested the feasibility of designing a tool to automatically generate items. For instance, [4] proposes creating a vocabulary test or exercise from a general corpus using a concordancer, and [5] suggests automatically generating CALL exercise from an electronic dictionary and a parsed corpus. Along the same line of research, we build AWETS, an automatic web-based English testing system that can greatly facilitate the creation of multiple choice vocabulary test. The system, designed with the central concern of adaptability, can generate multiple choice vocabulary test items in accordance with the conditions input by test administers. The system consists of three independent yet interrelated modules: the item generation module, the test delivery module, and the record keeping module.

2 The Item Generation Module

The system is developed based on a large collection of electronic texts and natural language processing tools such as a morphological analyzer and a part-of-speech tagger. The procedures of building the system are as follows.

1. Collection of a Text Database: We retrieve free electronic English texts from the internet primarily from Project Gutenburg and the Sinorama Magazine. Texts in Project Gutenburg are mainly literary works, while those in the Sinorama Magazine contain articles about the culture and events in Taiwan. To ensure that the retrieved texts are not too difficult for our learners, we only include works published after 1960.

The corpus size is about 0.2 million words.

2. Lemmatization: All the retrieved texts are processed by a morphological analyzer developed by University of Pennsylvania which changes regular and irregular inflections into their lemmas, i.e. basic forms (e.g. ran => run, happier => happy).
3. Frequency counts of lemmas: After lemmatization, frequency count of each lemma in the entire corpus is conducted.
4. Sorting of the frequency count of the lemmas in descending order:
5. Identification of the difficulty levels of each lemma: Three levels of difficulty are specified. They correspond to college entrance exams, TOEFL, and GRE. Each level has a range of adjustable values. At present, the range of these three values is stipulated as follows.

College Entrance Exam	Words which fall in the range of the most frequently occurring 3000 – 5000 lemmas
TOEFL	Words which fall in the range of the most frequently occurring 5001 – 7000 lemmas
GRE	Words which fall in the range of the most frequently occurring 7001 – 9000 lemmas

6. Tagging: Each text is processed by Eric Brill' s tagger which labels each word its part-of-speech information.
7. Indexing of each word: A database is created which records the documents and position in which a word occurs so that sentences containing a specified word can be retrieved in no time.

Test administrators can choose the level of difficulty, the part-of-speech of words, as well as the number of questions to be tested. Once the choices are made, the system will randomly retrieve sentences which meet the input conditions via the index. A subroutine then converts the retrieved sentences into multiple choice questions. The distracters of the questions are chosen from words of the same difficulty level as the target word. Figure 1 is the user interface for inputting conditions. Figure 2 is the automatically generated test items.

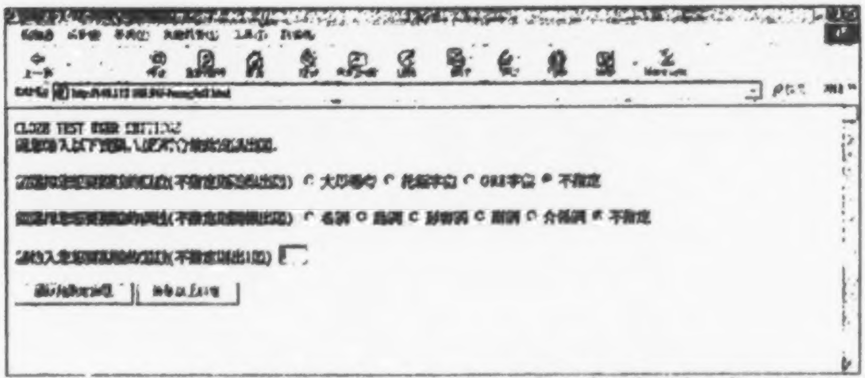

CLOZE TEST
○ 大學聯考 ○ 托福字彙 ○ GRE字彙 ● 不指定
○ 名詞 ○ 動詞 ○ 形容詞 ○ 副詞 ○ 介係詞 ● 不指定

Figure 1. User interface to choose difficulty level, part-of-speech, and number of questions

BEST COPY AVAILABLE

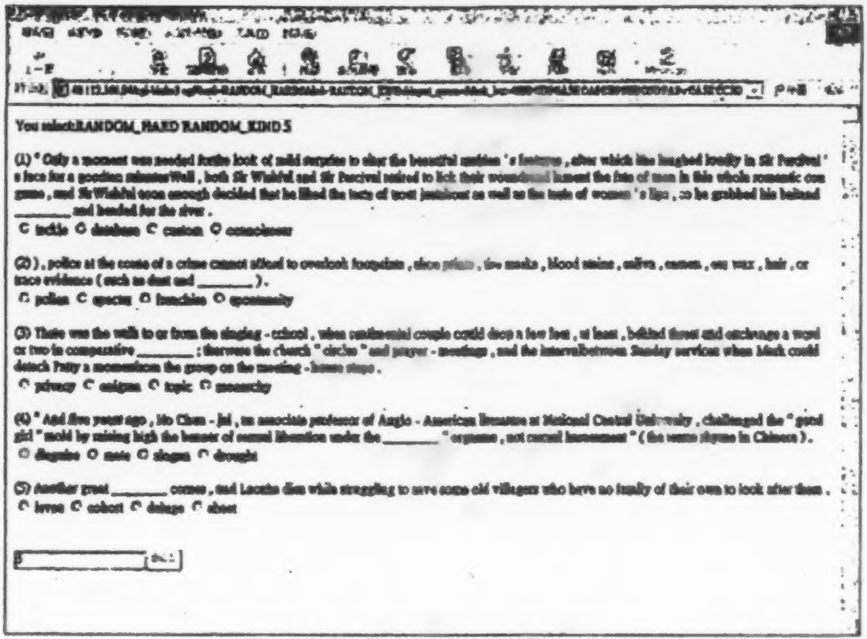


Figure 2. Test items generated by the system

As shown in Figure 2, the system is capable of generating individualized on-line vocabulary tests in the context of cloze tests based on the conditions input by a user. The system can thus be used as an excellent tool for self-paced vocabulary learning. If a learner wants to practice verbs at the TOEFL level, the system can create hundreds of such questions. As soon as he submits his answer, the system can check his answer and immediately present the correct answer to the user. Besides, if a test administer wants to change the difficulty level of the test, he can do it easily by changing the frequency range. To further facilitate the creation of vocabulary tests, the system also allows the test administer to decide which word should be tested. This is particularly useful for creating achievement tests. Once the tester inputs the words and the number of questions, the system can randomly generate multiple choice vocabulary tests in the context of cloze tests. Besides a corpus, AWETS also uses Wordnet, a lexical database developed at Princeton University, to generate items. It extracts the explanation of a lexical item and create multiple choice questions based on the item.

3 The Test Delivery Module

As described above, the item generation module can randomly create a specified number of questions in accordance with the input conditions by a test administer. To make test delivery more efficient, the test databank is created off-line. In other words, all the sentences meeting the input conditions are retrieved before the test starts. These sentences are converted into test items by a subroutine and then stored in the database. A subroutine then randomly retrieves a specified number of items from the databank and present them to the testees when the test starts. To ensure wide and unpredictable sampling, the subroutine is designed in such a way that no two tests are identical and no word will be tested twice in any test. The AWETS database also provides an interface (cf. Figure 3) for the test administer to input specification for the test. The interface allows the test administer to input the name of the test, the number of items, the time limit during which each question should be answered, and the number of times each testee can take the test. The test administer can further choose which classes and which words should be included in the test. After the test information is input by the test administer, testees proceed with the following procedures. They first input their user names and passwords. Before the real test begins, they are given 5 questions for practice. This procedure can help testees become familiar with the format of the questions. An interface and a test item such as Figure 4 is presented to the testees. As mentioned earlier, each question must be answered within a specified time limit. As soon as a question appears on the screen, the system begins to count down

BEST COPY AVAILABLE

the time left. The randomized questions and the time limit make cheating in the examinations much more difficult. Without these two functions, students might try to find answers from the person who sit next to them or from an on-line dictionary. The countdown device might also achieve a beneficial backwash, because testees need to speed up reading the question in order to finish the questions within the time limit.

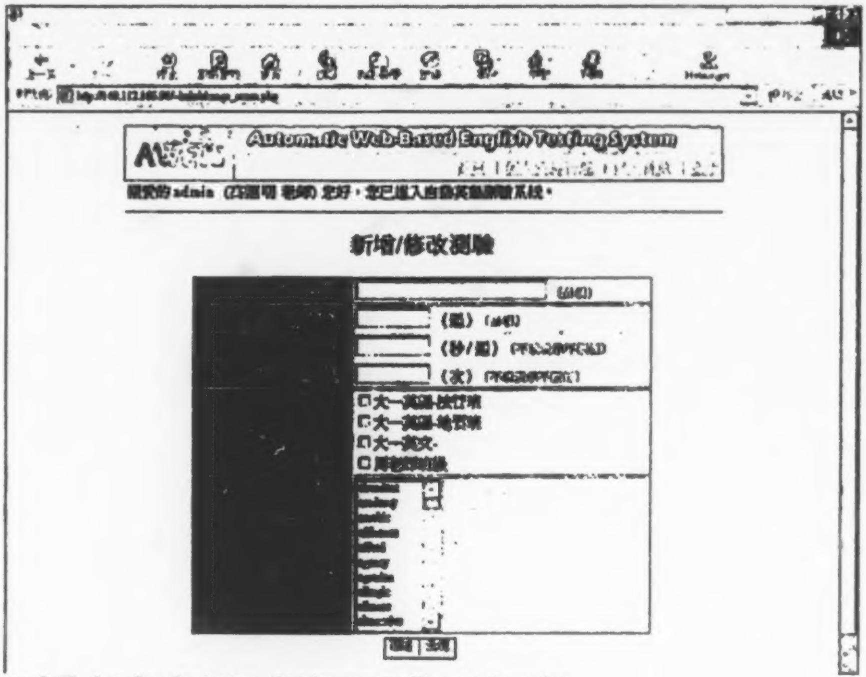
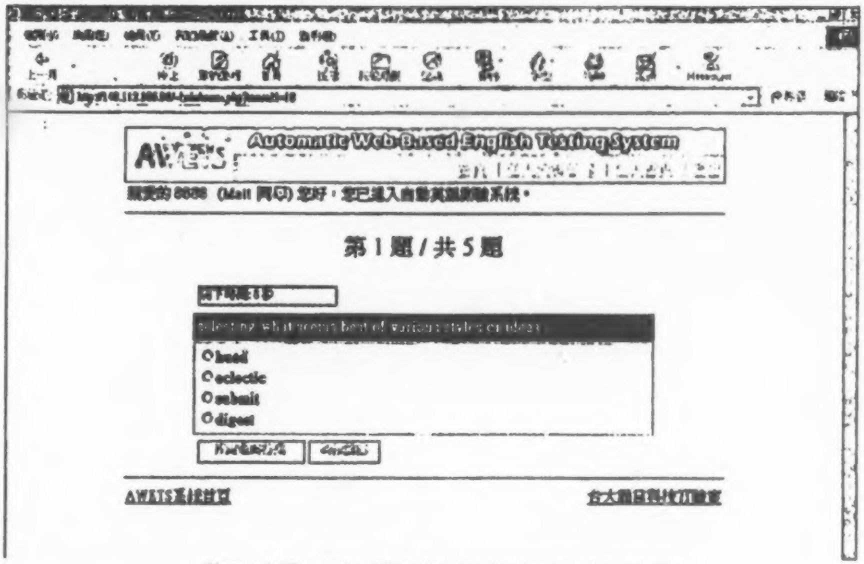

Automatic Web-Based English Testing System
新增/修改測驗

Figure 3. The interface for the test administer to specify test information


Automatic Web-Based English Testing System
第1題/共5題

selecting what seems best of various styles or ideas

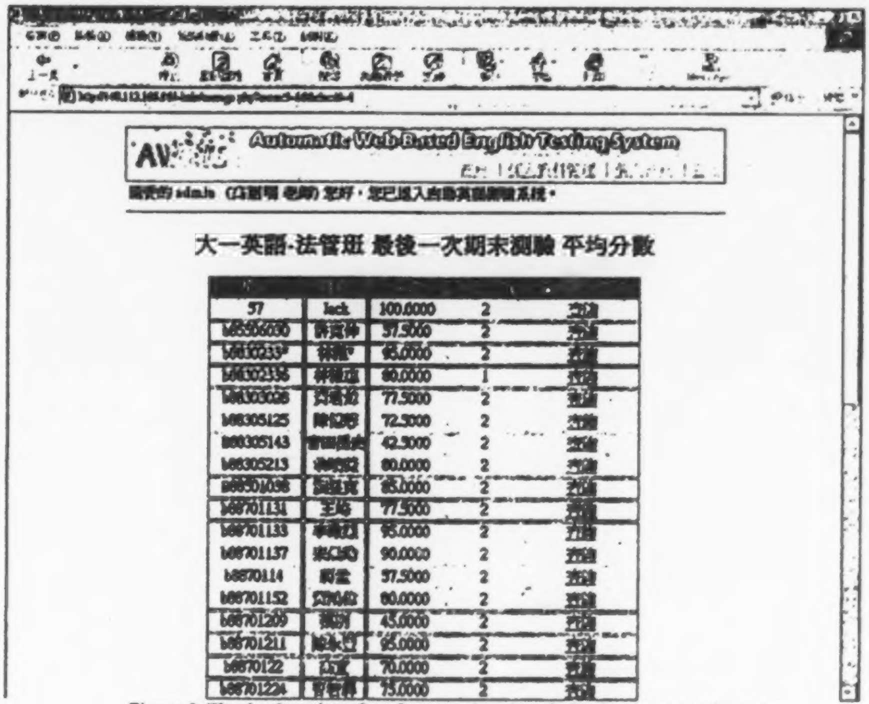
○ heed
○ eclectic
○ submit
○ digest

Figure 4. The testees' interface and a generated test item

BEST COPY AVAILABLE

4 The Record Keeping Module

After each test, the system records the registration number, the name of the student, the test id number, the name of the test, as well as the student' s score in each test. The database component allows teachers to query a student' s record or the whole class's scores in an exam via the interface in figure 5.



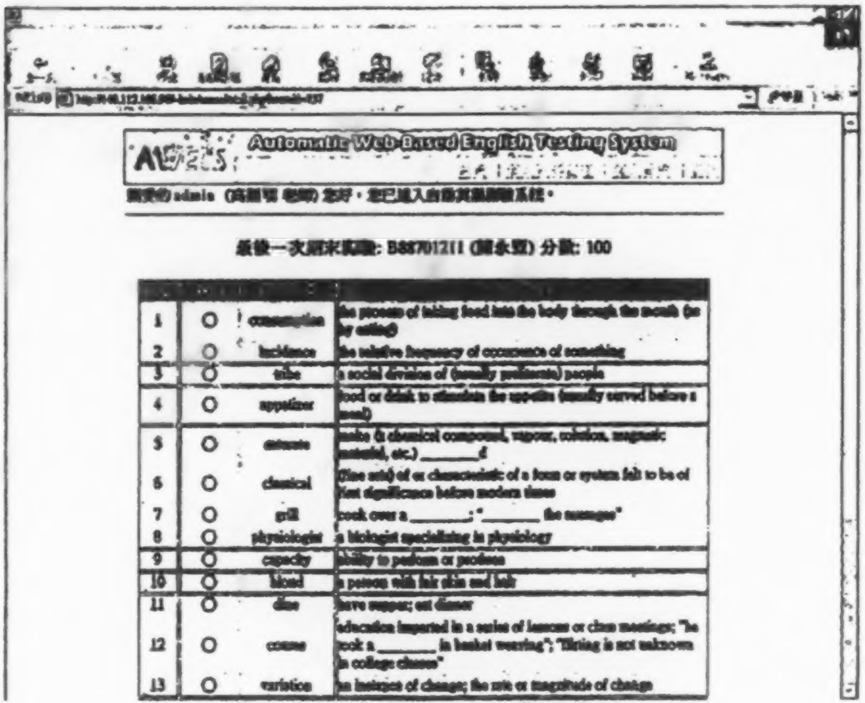
大一英語-法管班 最後一次期末測驗 平均分數

57	Jack	100.0000	2	[illegible]
[illegible]	[illegible]	37.5000	2	[illegible]
[illegible]	[illegible]	95.0000	2	[illegible]
[illegible]	[illegible]	80.0000	1	[illegible]
[illegible]	[illegible]	77.5000	2	[illegible]
b86305125	[illegible]	72.5000	2	[illegible]
b86305143	[illegible]	42.5000	2	[illegible]
b86305213	[illegible]	80.0000	2	[illegible]
[illegible]	[illegible]	85.0000	2	[illegible]
b88701131	[illegible]	77.5000	2	[illegible]
b88701133	[illegible]	95.0000	2	[illegible]
b88701137	[illegible]	90.0000	2	[illegible]
b8870114	[illegible]	37.5000	2	[illegible]
b88701152	[illegible]	80.0000	2	[illegible]
b88701209	[illegible]	45.0000	2	[illegible]
b88701211	[illegible]	95.0000	2	[illegible]
b8870122	[illegible]	70.0000	2	[illegible]
b88701224	[illegible]	75.0000	2	[illegible]

Figure 5. The database interface for querying the whole class' s scores in a test

The database component greatly facilitates the calculation of validity and reliability. When testees are given more than one set of test items in a given test, the correlation of the scores can be easily computed. The system also records all the questions and testees' responses. These data can be used to analyze testees' test-taking strategies. With this function, item analysis is possible although no test candidates have identical tests.

BEST COPY AVAILABLE



Automatic Web-Based English Testing System

最後一次期末測驗: B88701211 ([illegible]) 分數: 100

1	○	consumption	the process of taking food into the body through the mouth (as by eating)
2	○	incidence	the relative frequency of occurrence of something
3	○	tribe	a social division of (usually preliterate) people
4	○	appetizer	food or drink to stimulate the appetite (usually served before a meal)
5	○	saturate	make (a chemical compound, vapour, solution, magnetic material, etc.) ________d
6	○	classical	(fine arts) of or characteristic of a form or system felt to be of first significance before modern times
7	○	grill	cook over a ________; "________ the sausages"
8	○	physiologist	a biologist specializing in physiology
9	○	capacity	ability to perform or produce
10	○	blond	a person with fair skin and hair
11	○	dine	have supper; eat dinner
12	○	course	education imparted in a series of lessons or class meetings; "he took a ________ in basket weaving"; "flirting is not unknown in college classes"
13	○	variation	an instance of change; the rate or magnitude of change

Figure 6. A student's responses to the test items

5 Some Problems of AWETS

Although AWETS performs relatively well, there are some limitations which prevent it from being a completely reliable testing instrument. First, the basic assumption that difficulty of words can be determined by frequency is challenged by some scholars, since there are some words common in everyday life but much less common in texts. Moreover, a word might have several meanings some of which are much more difficult than the others. The approach proposed in this paper cannot distinguish the difficulty of the different meanings of a word. Another question is whether there might be more than correct answer in generated test items. When AWETS automatically creates multiple choice questions, it randomly chooses distracters from the dictionary. Although the distracters rarely fit the context, it might happen that some of them are acceptable. Note that choosing distracters with different parts-of-speech from the target word does not solve the problem, because a word might be used in different parts-of-speech. It should also be admitted that although AWETS can create individualized tests, it lacks a rigid method to ensure equal difficulty for all testees. Another technical problem involved is that the part-of-speech tagging program and the program which identifies sentence boundary is not one hundred percent correct. This might result in undesirable test items. Even when sentence boundary is correctly identified, some sentences might not be appropriate in testing a learner when taken out of context. This is particularly true of short sentences. Long sentences, however, are not always unproblematic. In a vocabulary test, all the words in the sentence are meant to give the contextual clues except the target word. In other words, the target word should ideally be the most difficult word in the sentence. Consequently, if there is a word in the same sentence more difficult than the target word, the test item might not be appropriate. Questions like these all require more rigid methods than those adopted in current implementation of AWETS.

6 Conclusion and Future Research

In this paper, we introduce AWETS, a web-based system that can automatically create vocabulary tests and

BEST COPY AVAILABLE

adapt items according to the conditions input by test administers. AWETS greatly facilitates the creation of vocabulary tests and has fully automated procedures for item generation, test delivery, scoring, and record keeping. At present, the validity and reliability of the automatically generated test items are being investigated. Future research will focus on solving the problems noted in section 5 by using sense-tagged texts and more rigid methods to identify difficulty of words.

Acknowledgements: The author wishes to thank Miss Lin Gui-Guang, Miss Liang Jing-Xiou, Mr. Zhao Zheng-Ming, and Mr. Lin Zheng-Ru for helping implement AWETS. AWETS is supported in part by a grant from the Computing and Information Network Center at National Taiwan University for the promotion of asynchronous learning.

References

[1] Bulter, Jonathan. Cloze procedures and concordances: The advantages of discourse level authenticity in testing expectancy grammar. System 19, 1/2, pp. 29-38, (1991).
[2] Hughes, Arthur. Testing for Language Teachers. Cambridge University Press, (1989).
[3] Stevens, Vance. Classroom concordancing: Vocabulary materials derived from relevant authentic text. English for Specific Purposes Journal 10: 35-46, (1991).
[4] Wilson, Eve. The Automatic Generation of CALL Exercises form General Corpora. In Wichmann et al. (eds.) Teaching and Lanuage Corpora. Longman, pp. 116 – 130, (1997).

Case-Based Evaluating Assistant of Novice Programs

Hiroyoshi Watanabe, Masayuki Arai and Shigeo Takei
School of Science and Engineering, Teikyo University
Toyosatodai Utsunomiya-shi Tochigi, 320-8551 Japan
E-mail: {hiro,arai,takei}@ics.teikyo-u.ac.jp

This paper presents a method of implementing an evaluating assistant system that supports teachers' evaluation work of students' programs using case-based reasoning. The target evaluation tasks are to judge whether a student's program satisfies the requirements of the given problem and to give advice for the student's program. The case-based evaluating assistant system compares a program submitted by a student with evaluation cases in the case-base. If some case matches the program, the system applies the judgment and advice on the case to the program. We implemented a case-based evaluating assistant system for novice programs written in an assembly language based on the proposed method. The implemented system was utilized for actual classes and the results showed that the system reduced the teachers' evaluation work drastically.

Keywords: **program evaluation, programming classes, supporting teachers, case-based reasoning.**

1 Introduction

This paper describes a method of supporting teachers' evaluation work of students' programs. In programming education, programming exercise courses play an important role, because writing programs is indispensable to learning programming. In programming exercise classes, however, the teacher's loads of evaluation tend to be very heavy because the teacher has to read so many programs and reports. We aim to implement a computer system as an evaluating assistant that supports the teachers' evaluation work.

There are two approaches to the evaluation of students' programs. The first one is to diagnose programs and give advice by knowledge-based program recognition [1]-[3]. Most of the systems based on the approach require a huge amount of knowledge on bugs, and it would be difficult to constitute the systems practically. The second approach is to support teachers' evaluation work [4]. This approach may not necessarily aim at automating the evaluation work. It aims at implementing practical systems by limiting computers' evaluation work. We took the second approach.

We consider the program evaluation work as collaboration between teachers and computer systems and propose an evaluating assistant model of programs. We also propose a framework of the case-based evaluating assistant system.

2 Target Task and Evaluating Assistant

2.1 Task of Program Evaluation

The target evaluation tasks of a student's program are the following two tasks: (1) the first task is judging whether a student's program satisfies requirements of the given problem. When teachers set problems, they have educational intentions about what students should learn, namely concepts, algorithms, instructions and so on. Teachers read students' programs to see whether the educational intentions are achieved. Therefore, teachers accept a student's program when the program satisfies requirements of the given problem. The first task is defined on the assumption that students have to submit their programs over and over until their programs are accepted. (2) The second task is giving written advice. Teachers give advice to students whether they accept a program or not: teachers give advice about the reasons why the program is rejected, and advice about bettering the program even if the program is accepted.

2.2 Evaluating Assistant of Programs

Figure 1 illustrates an evaluating assistant in the electronic submission environment of programs. The evaluating assistant pre-evaluates submitted programs and a teacher can refer to the results when he or she evaluates the programs. If the teacher trusts the evaluating assistant, the results from the assistant can be sent to students directly. Such an evaluating assistant is expected to save a teacher a lot of time and energy.

The output of the evaluating assistant consists of evaluation results, their reasons and the degree of confidence. The evaluation results include the judgment of acceptability (accept or reject) and written advice. The degree of confidence is one of surely, probably or unknown. When the degree of confidence is unknown, evaluation results and their reasons are not given.

The evaluation results of the assistant are required to be always correct when the degree of confidence is surely. If so, the results of the assistant with surely confidence can be sent to students directly, in other words, teachers can trust the evaluating assistant.

The evaluating assistant should have the capability to learn. The final results of the teacher's evaluation are available for the learning. If the assistant is capable of learning, almost the same programs as ones which the assistant has evaluated incorrectly, are expected to be evaluated correctly in the future.

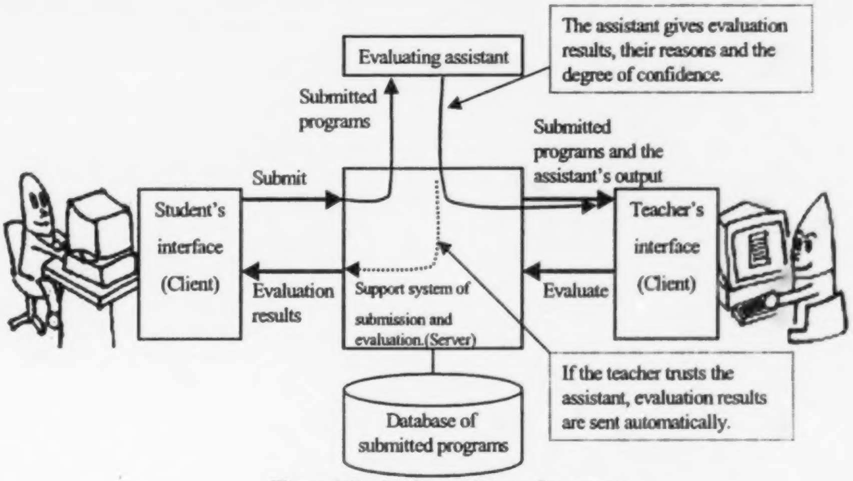

The assistant gives evaluation results, their reasons and the degree of confidence.
Evaluating assistant
Submitted programs
Submitted programs and the assistant's output
Submit
Student's interface (Client)
Evaluation results
Support system of submission and evaluation.(Server)
Evaluate
Teacher's interface (Client)
If the teacher trusts the assistant, evaluation results are sent automatically.
Database of submitted programs

Figure 1 Evaluating assistant of programs

3 Case-based Evaluating Assistant

The case-based reasoning approach is one of the best approaches to implement an evaluating assistant described in Section 2.2. Case-based reasoning systems make use of stored past cases directly in solving newly presented problems [5]. The case-based evaluation of programs is defined as "if some evaluation case of a program whose implementation is the same as the newly given program, then the evaluation results on the case are applied to the given program".

3.1 Representation of Cases

A case for the case-based evaluating assistant consists of retrieval information, problem description, solution description and maintenance information.

(1) The retrieval information includes problem identification that the program is written for and features of the program. The features of the program depend on target programming languages. For example, numbers of if-statements, while-statements and other statements are available in the case of language C.
(2) The problem description in the domain of a program evaluation task is a program list itself. A program list should be represented as a normalized form [2], or a generalized form [6], because there are many variations of program lists for the same implementation.
(3) The solution description includes the judgment of acceptability and written advice.
(4) The maintenance information includes a teacher's name and the date of adding or updating the case.

3.2 Processes of Case-based Program Evaluation

The processes of case-based program evaluation are the following:
(1) Problem analysis: The retrieval information is extracted from a student's program list.
(2) Case retrieval: Cases are retrieved using information generated by analyzing a given program. Cases that have no possibility of matching the given program should be pruned here.
(3) Evaluating and selecting cases: Evaluating cases is the process of matching a given program against cases. The purpose of the process is to investigate whether the given program has the same implementation as the cases, or not. All candidates of cases are evaluated and the best match case is selected. The method of matching programs depends on the target programming languages.
(4) Applying and adapting cases: If there is a case that matches the given program, the judgment of acceptability on the case is applied to the given program. In addition, advice sentences on the case are available for the given program, although the sentences should be adapted for the given program. If no case matches the given program, the judgment and advice is not generated.

3.3 Case-base Maintenance

The maintenance of the case-base is performed using teacher's final evaluation results. One of the most important maintenance tasks is adding new cases when the evaluation results of the assistant are different from the teacher's. New cases are also added when the confidence of the evaluating assistant is not *surely*. More advanced maintenance, e.g., generalizing, specializing and forgetting cases [7], may be needed in order to refine the case-base.

4 The Evaluating Assistant System for Assembly Language Programs

Based on the proposed idea, we implemented a case-based evaluating assistant system for novice programs written in an assembly language [6]. The target assembly language is CASL which is adopted in examinations for information-technology engineers certified by the Japanese ministry of international trade and industry.

4.1 Implementations Depending on The Target Language

In this section, implementations depending on the target language CASL are described.
(1) Evaluating the program's action: Before the case-based program evaluation, the assistant system tests the action of a submitted program using prepared sample data. Only programs executed correctly are evaluated by case-based reasoning [6].
(2) Case representation: Although cases are represented in the form described in Section 3.1, no features except for a program ID are used for the retrieval information. A program list in a case is represented in CASL itself, or its generalized form that we defined [6].
(3) Case retrieval: The implemented system retrieves all cases whose problem ID is the same as a given program. That is to say, the system does not prune candidate cases.
(4) Case evaluation (program matching): The program matching process aims at making consistent correspondences of instructions, labels and registers between a case and a student's program [6]. If the following condition1 is met, the case matches the given program.

- **Condition 1**: All instructions of the case correspond to instructions of the given program, and all instructions that correspond to nothing do not affect to the program's action.
  Especially, if the following condition is satisfied, it is called a "perfect match":
- **Condition 2**: Instructions of the case and the given program correspond one-to-one and the differences of the order of corresponding instructions are trivial.

If the best match case meets condition2, *surely* is assigned as the degree of confidence. If the best match case meets condition1 but not condition2, *probably* is assigned. In the other cases, that is, when no case meets the condition 1, *unknown* is assigned.

4.2 Experimental Results

The implemented assistant system was utilized for actual classes of the CPU and assembly language course at our university in 1999. Seventy-three sophomore students in the department of computer science took this course. Problems presented in classes of the course are the following: (P1) select bigger of the two given integers, (P2) sum the given N integers, (P3) select the maximum of the given N integers, (P4) rotate N bits to the right and (P5) check the correspondence of "(" and ")".

Table 1summarizes results of using the assistant system. The following are found from Table 1:

- The values of (f) show that the implemented case-based assistant system achieves sufficiently high accuracy of judgments. Furthermore, the accuracy of the case-based assistant system satisfies the requirements described in Section 2.2, because it is a hundred percent in cases of *surely* confidence.
- The values of (g) show that the ratios of available advice without modifying are not as high as the accuracy of judgments, although it is fairly high.
- Because teachers do not need to evaluate the acceptability when the case-based assistant system outputs evaluation results with surely confidence, it is estimated that the system reduces the teachers' evaluation work by percentages shown as (h). In other words, by using the assistant system, the evaluation work of teachers is reduced by 60 to 90 percent depending on problems.

These results demonstrate that the case-based assistant system is very effectual in reducing teachers' evaluation work. Still, there is room for improvement in the capability to generate written advice.

Table 1 Evaluation data of the assistant system based on practical use in classes

	P1	P2	P3	P4	P5
(a)Cases saved in the case-base	15	10	29	34	38
(b)Submitted programs	119	119	140	156	157
(c)Programs rejected by checking their action	44	39	44	54	79
(d)Programs evaluated by the assistant system with *surely* confidence	62	72	72	67	46
(e)Programs evaluated by the assistant system with *probably* confidence	8	3	5	5	0
(f)Judgment accuracy of the assistant system (%)	100 (100)	100 (100)	97.4 (100)	100 (100)	100 (100)
(g)Ratio of available advice generated by the assistant system without modifying (%)	92.9 (100)	69.3 (72.2)	88.3 (90.3)	66.7 (64.2)	91.3 (91.3)
(h) $((d)/((b)-(c))\times 100$ (%)	92.7	90.0	75.0	65.7	59.0

( ) : values for programs evaluated by the assistant system with *surely* confidence only.

5 Conclusions

We have proposed a concept of a program evaluation assistant and a method of implementing the assistant by case-based reasoning. Based on the method, we implemented a system for a simple assembly language CASL and used it in actual classes; the results demonstrated that the system reduced teachers' evaluation work drastically. We plan to improve the modification functions of advice sentences (written advice) and the method of the case-base maintenance. This research was supported in part by the Japanese Ministry of Education Grant No.11680400 and No.12780293.

References

[1] Adam,A. and Laurent,J.P., "LAURA, A System to Debug Student Programs, Artificial Intelligence", vol.15, pp.75-122 (1980).

[2] Johnson,W.L., "Understanding and Debugging Novice Programs", Artificial Intelligence, vol.41, pp.51-97 (1990).

[3] Ueno,H., "Concepts and Methodologies for Knowledge-Based Program Understanding – The ALPUS's Approach", IEICE Trans. Inf. & Syst., vol.E78-D, no.2, pp.1108-1117 (1995).

[4] Konishi,T., Suyama,A. and Itoh,Y., "Evaluation of Novice Programs Based on Teacher's Intention", Proc. of ICCE95, pp.557-566 (1995).

[5] Kolodner,J. ," Case-Based Reasoning", Morgan Kaufmann Publishers,Inc. (1993).

[6] Watanabe,H., Arai,M. and Takei,S. , "Automated Evaluation of Novice Programs Written in Assembly Language, Proc. of ICCE99, vol.2, pp. 165 - 168 (1999).

[7] Watanabe,H., Okuda,K. and Yamazaki,K., "Methods for Adapting Case-bases to Environments", IEICE Trans. Inf. & Syst., vol.E82-D, no.10, pp.1393 - 1400, (1999).

CBR-TUTOR: A Case-Based Reasoning Approach to an Internet Agent-based Tutoring System

Rhodora L. Reyes and Raymund Sison
ccsrlr@ccs.dlsu.edu.ph, ccsrcs@ccs.dlsu.edu.ph
College of Computer Studies, De La Salle University. Professional Schools Inc.
Manila, Philippines

Providing individualized instruction is an important tutoring task. Different learners have different needs. This task becomes more important when dealing with learners on the web. This paper presents the CBR-TUTOR, an Internet-based tutoring agent system that uses case-based reasoning approach in providing adaptive instruction to its learners. Using CBR in the tutor model enables the tutor to reference from past experiences and identify which instructional strategies were successful given a similar situation or student characteristics. The CBR-TUTOR is designed as a distributed problem solving architecture where each agent performs decision-making tasks and cooperates to help improve the effectivity of the tutoring system.

Keywords: **Intelligent Tutoring Systems, Case-Based Reasoning, Web-based learning, Agents**

1 Introduction

"The Internet now provides a possible new dimension to information technology in education, not only in terms of potential as a vast information resource, but also in respect of interaction and knowledge construction between individuals" (Wood, 1999). Unfortunately, most of the learning systems and electronic textbooks accessible on the web lack the capabilities of individualized instruction and user-adapted learning support that are emergent features of Web-based Intelligent Tutoring Systems [10].In providing individualized instruction, it is important to diagnose the problems of individual learners and identify how to adapt instruction and remediation to the needs of the individual learner. The diagnosis of the learner often involves analysis of learner errors. This is the task of the student model component of an Intelligent Tutoring System (ITS). A student model is an approximate representation of a student' s knowledge about a particular domain, which accounts for the students' solutions to given problems [9]. The tutor model component of an ITS, on the other hand, uses the student model to determine how to provide instruction and remediation [6].

The tutor model must be able to recognize the similarity and differences in the needs of these learners. Different learners have different needs. The task becomes more important when dealing with students on the web. Therefore, Internet-based tutoring systems must be able to reference to past experience in order for it to know which approach will be appropriate given a situation (or case). However, no past situation is ever exactly the same as a new one and domain knowledge for instructional strategies is oftentimes incomplete. This makes the tutor model incapable of using the instructional method appropriate to the learner. It is therefore necessary to create a tutor model that has the capability to understand new situations in terms of old experiences and adapt an old solution to fit a new situation. Cased-Based Reasoning (CBR) suggests a model of reasoning that incorporates problem solving, understanding, and learning; and integrates all with memory processes. CBR can mean adapting old solutions to meet new demands, using old cases to critique new solutions or reasoning from precedents to interpret a new situation or create an equitable solution to a new problem. [5]. CBR cycle has four phases: *retrieving* the most similar case or cases, *reusing* the information and knowledge in the case to solve the problem, *revising* the proposed solution and *retaining* the parts of the experience that is likely to be useful for future problem solving [2].

Using CBR in the tutor model enables the tutor to reference from past experiences and identify which instructional strategies where successful given a similar situation or student characteristics. Existing

Case-based Intelligent Tutoring Systems use cases for teaching the learners the domain, that is, they use cases as pedagogy similar to the way exercises are used as strategy for teaching. Examples of such systems are Case-based Intelligent Tutoring Systems for operators of dynamic systems (CB-ITS) [3], Georgia Tech Case-Based Intelligent Tutoring Systems for Pilots (GT-CBITS) and Case-Based Reasoning Approach to Simulation-Based Intelligent Tutoring Systems for Tactical Action Officers (TAO ITS) [7]. However, none of these systems use CBR as an approach in helping the tutor identify similar experiences encountered when providing individualized instruction to the learners.

CBR-TUTOR is an Internet agent-based tutoring system that uses the CBR approach in providing adaptive instruction to its learners. It is designed as a distributed problem solving architecture where each agent performs decision-making tasks and *cooperates* to help improve the effectivity of the tutoring system. Cooperative agents are agents that are assigned to do specialized tasks to solve a common goal. Section 2 of this paper discusses the architecture of CBR-TUTOR followed by the discussions of its components in Section 3. Finally, the conclusion and future works will be presented.

2 CBR-Tutor Architecture

CBR-TUTOR is an Internet-based tutoring agent system that uses case-based reasoning (CBR) approach to determine how to provide individualized instruction to its learners. This section discusses the architecture of the CBR-TUTOR in terms of its components and their relationships.

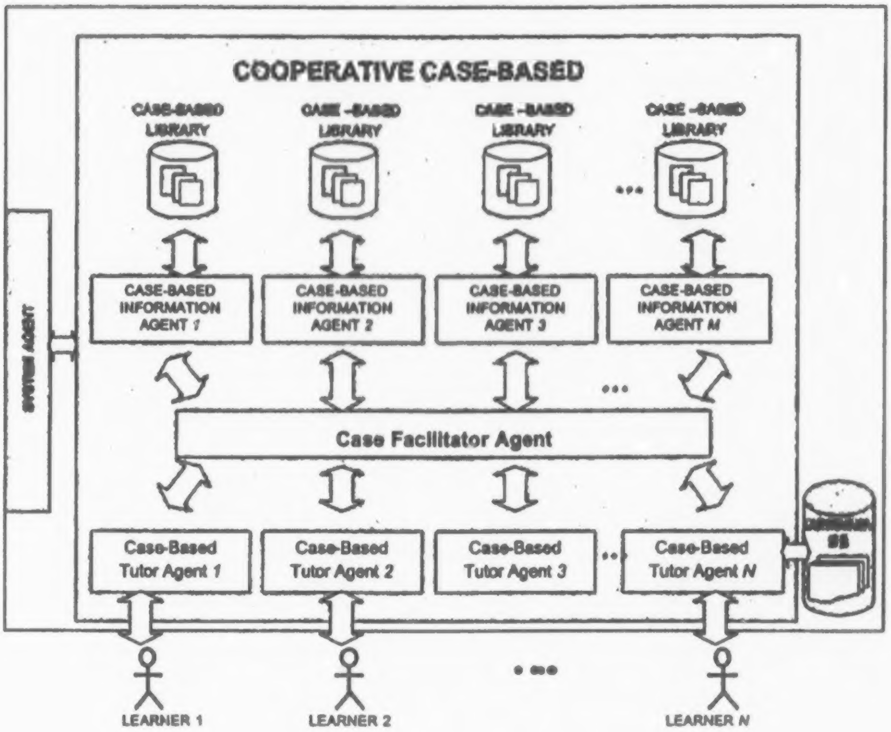
CBR-TUTOR is a distributed problem solving (DPS) system comprised of the system agent (SA), cooperative case-based module (CCBM) and the curriculum database (CDB). Figure 1 shows the architecture of CBR-TUTOR.

The SA serves as the registry module that contains the complete list of all agents initiated in the system. Whenever there is an unregistered learner (i.e., first time user of the system), the SA initializes the agent (or agents) that will be used for tutoring the learner and informs the CCBM about it. Whenever necessary, the SA also decides if there is a need to process the requests of creation of new agents in the system by CCBM.

The cooperative case-based module (CCBM) is the core component of the CBR-TUTOR. It is composed of cooperating agents designed for actual tutoring of the learners, retrieval, filtering, indexing and learning of cases, and facilitation of requests from different agents in the system. Its sub-components are case-based tutor agents (CTAs), case facilitator agent (CFA), case-based information agents (CIAs), and case-based libraries (CBLs). The CTAs are the actual tutors assigned to the learners while CIAs are agents whose tasks are to retrieve, filter, index and modify (or learn) cases in its CBL. A CBL contains the set of cases used in the system. Discussion of the CCBM components is discussed in detail at section 3.2.

Depending on the planned teaching activities, a CTA accesses the curriculum database (CDB) for content presentations. These presentations include the lessons, examples, elaboration, exercises, answers, definitions and descriptions.

Figure 1. The CBR-TUTOR Architecture


COOPERATIVE CASE-BASED
CASE-BASED LIBRARY
CASE-BASED LIBRARY
CASE-BASED LIBRARY
CASE-BASED LIBRARY
CASE-BASED INFORMATION AGENT *1*
CASE-BASED INFORMATION AGENT *2*
CASE-BASED INFORMATION AGENT *3*
CASE-BASED INFORMATION AGENT *M*
SYSTEM AGENT
Case Facilitator Agent
Case-Based Tutor Agent *1*
Case-Based Tutor Agent *2*
Case-Based Tutor Agent *3*
Case-Based Tutor Agent *N*
LEARNER 1
LEARNER 2
LEARNER *N*

3 CBR-Tutor Components

The primary components of the CBR-TUTOR are its specially designed agents. These are COOPERATIVE CASE-BASED MODULE (CCBM) and SYSTEM AGENT (SA). This section discusses the detailed discussion of these components.

3.1 Cooperative Case-Based Module

The COOPERATIVE CASE-BASED MODULE (CCBM) is the heart of the CBR-TUTOR. It is composed of specialized agents that have specific decision-making task that cooperates to help provide individualized and adaptive instruction using the case-based reasoning approach. The major components of CCBM are CASE-BASED TUTOR AGENTS, CASE FACILITATOR AGENT, CASE-BASED INFORMATION AGENTS and CASE-BASED LIBRARIES, as illustrated in Figure 1.

3.1.1. Case-Based Tutor Agent

The CASE-BASED TUTOR AGENT (CTA) interacts directly with the learner. It creates a profile of its learner (i.e., learner type, lessons taken, performance, strategies applied, output from *external student modeling system*, etc.) and this information for evaluating the current scenario. (i.e., case). The CTA uses the case-based reasoning approach in planning for the teaching strategy to use, with the help of the other agents in CCBM.

The CTA has four components: INTERFACE, CASE-BASED MODULE, INSTRUCTIONAL PLANNER, and STUDENT KNOWLEDGE BASE, as shown in Figure 2.

BEST COPY AVAILABLE

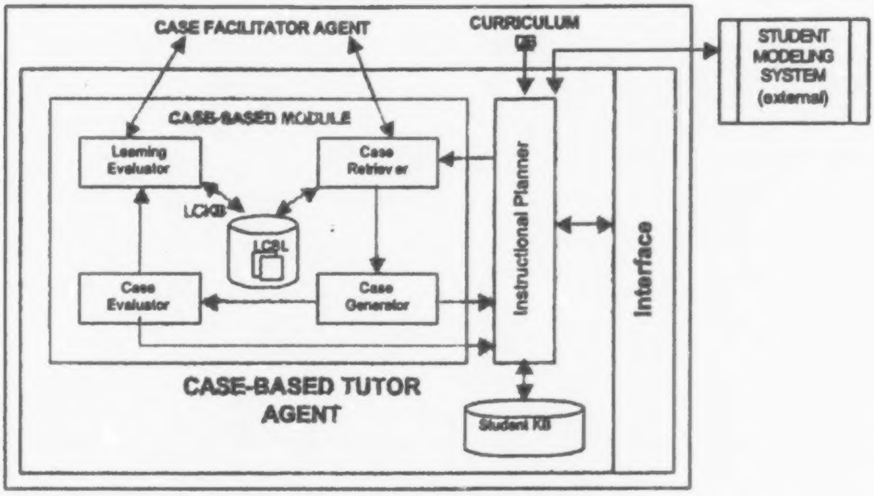


Figure 2. Components of a Case-Based Tutor Agent (CTA)

All communications between the CTA and the learner is done through the *interface*. The *interface* is designed as a *web-based interface* for simplicity and universal access. The system can be used through standard browsers.

The INSTRUCTIONAL PLANNER (IP) is the component of the CTA that assesses the current case (i.e., current teaching scenario), and communicates with the interface and the CASE-BASED MODULE (CBM). Based on the current case, the IP forwards its requests to the CBM for generation of helpful case (or cases). The proposed solution (i.e., result of the request) from the CBM is implemented by the IP. Implementation of the proposed solution requires the generation of content presentation. The IP does this by accessing the CURRICULUM DATABASE (CDB), which contains the lessons, exercises, examples, description, elaboration, answers and definitions.

The IP is also responsible for updating the STUDENT KNOWLEDGE BASE (SKB). The SKB contains information about the user including the name, password, identification number, and student type (i.e., low, medium, high). It also contains the result of the diagnoses of the *external* student modeling system, lessons taken, and performance evaluation of the leaner.

The component of the CTA that uses the Case-Based Reasoning (CBR) approach is the CBM. The CBM's tasks are to retrieve useful case (or set of cases), propose a solution (i.e., teaching plan), test and evaluate the proposed solution, and if needed, learn a new case. The CBM keeps local copies of cases that are frequently used during the tutoring sessions and stores it in the LOCAL CASE-BASED LIBRARY (LCBL). CBM also has a CASE RETRIEVER module, which retrieves cases from the LCBL and/or requests for cases through the CASE FACILITATOR AGENT (CFA). The retrieval approach used by the CASE RETRIEVER is the same as the retrieval approach used by the CIA (discussed in section 3.2.3.). The CASE RETRIEVER, depending on its *certainty factor*, decides whether to request for case retrieval through the CFA or use only the retrieved cases from the LCKB. The *certainty factor* is a measure used to evaluate the appropriateness or availability of cases in the LCBL. If the CTA does not have enough cases to solve the current case, it makes a request to the CFA for retrieval of *good* cases. *Good cases* are those that have potential to make relevant prediction about the new case [4]. This means that cases retrieved either helps the CTA achieve a goal or warns about the possibility of a failure or point out an unforeseen problem [5].

The cases retrieved (or requested) by the CASE RETRIEVER is given to the CASE GENERATOR, which in turn checks which of the retrieved case matches exactly the retrieved case. In the event of finding an exact match, the CASE GENERATOR proposes a solution based on the retrieved case and forwards it to the IP for *reuse*. However, it is seldom that previous case matches the current case exactly. It is therefore necessary for the CASE GENERATOR to revise the retrieved case and propose a new solution. The *proposed new solution* is tested and, if needed, repaired by the CASE EVALUATOR until a *confirmed solution* has been achieved. The

BEST COPY AVAILABLE

confirmed solution is then forwarded to the IP for implementation. The CASE EVALUATOR also forwards the *confirmed solution* to the LEARNING EVALUATOR for possible learning of the new case. If there is a need to learn a new case, the LEARNING EVALUATOR updates the LCBL and informs the CASE RETRIEVER about the changes in the LCBL. The approach for learning a new case (i.e., case library update) used by the learning evaluator is the same as the approach used by the CIA (discussed in section 3.2.3.). The LEARNING EVALUATOR also informs the CFA that a new case has been learned.

3.1.2 CASE FACILITATOR AGENT

The case facilitator agent (CFA) serves as mediator between the case-based tutoring agents (CTAs) and the case-based information agents (CIAs). This means that the CFA performs matchmaking of services thatcan be provided by CIAs and requests made by CTAs. The CFA tracks all requests for retrieval of cases and monitors the updating (i.e., learning) of new cases. It receives requests from CTAs, sends these requests to the candidate CIA (or CIAs) and returns responses to the requesting CTAs. The CFA has two major components: service request module, and agent information manager (see Figure 3).

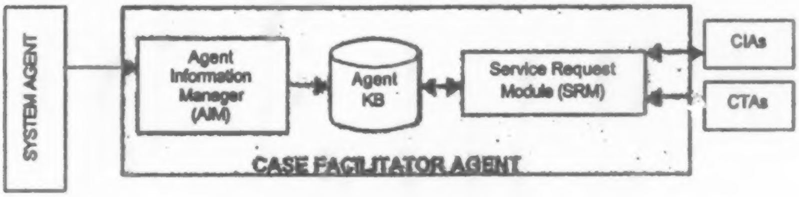


Figure 3. Components of the Case Facilitator Agent

The SERVICE REQUEST MODULE (SRM) supervises all requests from the CTAs. There are two types of requests: retrieval of cases and updating (learning) of new cases. Using the *knowledge* about the capabilities of the CIAs, the SRM performs matchmaking and assesses which CIAs are suitable to process the request. This knowledge includes the *indexing vocabulary*, *specialization* and *taxonomy of indexes* of each CIA. An *indexing vocabulary* is a set of relevant descriptors used to describe and index cases while *specialization* refers to the specific indexes being monitored by the CIA. The *taxonomy of indexes* contains information about the organizational structure of the indexes.

The SRM maps the capabilities of each CIAs to the requests and forwards the requests to the candidate CIAs. The SRM collects the result from all CIAs that responded to the request, checks and eliminates redundant cases before forwarding the result to the requesting CTA.

The AGENT INFORMATION MANAGER (AIM) monitors all the agents registered by the SYSTEM AGENT (SA) in the system. It has the knowledge of the *indexing vocabulary* and *specialization* of each CIA and the CTAs that have been registered. It also supervises the mapping and updating of the taxonomy of indexes. The knowledge about each agent and the taxonomy of indexes are used by the SRM to determine relationships, similarities and differences of the indexes of each CIA. This helps SRM in matchmaking the CTAs request and CIAs capability to process the request.

3.1.3 Case-Based Information Agent

The CASE-BASED INFORMATION AGENT (CIA) performs the tasks of retrieving cases, evaluating and filtering the retrieved cases in the CCBM. Each CIA has an associated CASE-BASED LIBRARY (CBL) for which they are responsible to maintain. Each CIA focuses on particular collection of features (i.e., dimensions) of the case. A *feature* is an attribute-value pair used in the description of the case [5]. This facilitates faster indexing, restructuring, searching and learning of cases. No two CIAs are exactly the same and despite similarities in their structure, they may return different results. A CBL of a CIA may contain cases that are similar to other CBLs (i.e., overlaps) or it may be totally different from the other CBLs .A CIA can also request the SYSTEM AGENT (SA) for *load reallocation* (i.e., creation of a new CIAs), if it is overloaded.

BEST COPY AVAILABLE

Retrieving Cases

Each CIA uses the combination of searching and matching when retrieving cases. The quality of the search algorithm is closely related to the quality of the organizational structure of the cases. The organizational structure of each CBL is designed, as a *flat library of cases* where cases are stored as simple lists (or array/files). All CIAs who were requested to retrieve cases use the *parallel retrieval approach*. In this approach, each candidate CIA will search for cases and forwards the result to the CFA (if there is any). The CFA will then be responsible for collecting and checking of results for redundancy.

Each CIA uses a SERIAL- SEARCH- PARTIAL- MATCH (SSPM) algorithm (outline shown in Table 1). Since each CIA *specializes* on specific *dimensions* (or set of features), it only searches a relatively small number of cases and searching is not expensive. A *dimension* collectively refers to all descriptive attributes of a case. In the SERIAL SEARCH algorithm, the entire CASE-BASED LIBRARY (CBL) maintained by the CIA is searched. This means that the accuracy of retrieval is a function only of how good the *matching functions* are. The matching function used by a CIA is the PARTIAL- MATCHING FUNCTION. In this approach, cases are indexed using observable features and derived features that capture partial similarities. A *combination of heuristic and numerical evaluation function* is used to compute for the matching and ranking of cases. The *heuristic function* filters cases that had mismatches in important features before comparing cases for their degree of similarity. To measure the degree of match of each pair, the Cognitive System' s (1992), evaluation function is adapted (see Equation 1).

$$\boxed{\frac{\sum_{i=1}^{n} w_i(sim\ (f_i^I, f_i^r)}{\sum_{i-1}^{n} w_i}}$$

Where w_i is the weight of the importance of dimension (slot) *i, sim* is the similarity function for $f_i^I and f_i^r$ primitives and are the values for the feature f_i in the input and retrieved cases, respectively.

The degree of match is represented as numerical values between 0 to 1. Closer matches have value closer to 1. Similarly, the similarity function with a value closer to 1 means that the features have high degree of similarity.

Equation 1: Evaluation Function

Table 1. Outline of SERIAL SEARCH- PARTIAL MATCHING ALGORITHM (SSPM)

1. For every case in memory, partially match input case:
   - Identify observable features and derived features
   - Compute for the degree of match by using the combination of heuristic and numerical evaluation function. The heuristics identifies the important criteria and then the numeric evaluation function is used for matching and ranking.
2. Return all the best case(s).

Learning Cases

Each candidate CIA decides whether the new case should be learned, and which information from the case to retain, in what form to retain it and how to index the case for later retrieval from similar problems. Learning is a natural consequence of using CBR. It learns by accumulating new cases and indexing the cases properly. A CIA learns basically by applying adaptation (or accumulating generalizations) to the cases and re-indexing of cases that are already in its CASE-BASED LIBRARY (CBL). Re-indexing is done when the case is recalled and it can not be used or it is used but results in failure. When a CIA learns a new case, it informs the CFA of the changes in its *indexing vocabulary* if there is any.

Indexing Cases

The indexing of a case indicates when a case should be retrieved. Cases are indexed based on the goal, solutions method and combinations of descriptors responsible for choice of particular solution. This type of indexing helps the CIA generate a case whenever there is a need to solutions to the current problem (e.g., what will be the strategy given the current case). Cases are indexed by CIAs by adapting the Universal Index Frame [8], which is a generally applicable descriptive vocabulary. UIF covers a broad range of domains about the interactions between agents and its goal.

Load Reallocation

Each CIA also has the capability to request the SA for a creation of a new CIA. When a CIA sees that it is already monitoring a large amount of indexes, using a *load factor*, the CIA can request the SA to divide the load by initializing a new CIA (or set of CIAs). A load is divided according to logical divisions and dimensions of the cases. A *load factor* is a measure of how many indexes a CIA can monitor without affecting the balance of the load of each CIA. The SA will then notify the CFA of the newly created CIA including the knowledge about it, and the changes in the knowledge about the requesting CIA.

3.1.4 Case-Based Library

The *CASE-BASED LIBRARY* (CBL) contains the set of stored cases. Cases represents specific knowledge tied to specific situations, it makes explicit how a task is carried out or how a piece of knowledge was applied or what particular strategies were used for tutoring the learners effectively. The CBL is designed as *flat library of cases* where cases are stored in a simple list. Since each CIA specialized on specific dimension of the case, the CBL is designed to be simple for faster accessing of cases.

Each case in the *CRC* has three major parts: *situation description, solution* and *result*. The *situation description* describes the goal (or set of goals), constraints on the goals, and other features of the problem situation. The *solution* part of the case contains the steps used to derive the solution (i.e., tutoring plan of action) and the justifications for decisions that were made. Alternative solutions and/or unacceptable solutions are also included in the *solution* part, if any. Finally, the *result* part of the case contains information about the success or failure of the solution, the explanation for failure or success, the repair strategy and the result of applying the repair.

3.2 System Agent

The SYSTEM AGENT (SA) contains the complete list of all agents initiated in the system. It verifies agent identities and provides their location in the network and transport addresses to the CASE FACILITATOR AGENT (CFA). It also stores additional information about the status of the agent and its type.

The SA communicates directly to the COOPERATIVE CASE-BASED MODULE (CCBM) and performs the following functions:

- Determines if the learner is an unregistered learner, initializes and assigns a CASE-BASED TUTOR AGENT (CTA) for the unregistered learner
- Processes requests from CASE-BASED TUTOR AGENTS(CIAs) for creation of a new CIA
- Monitors the complete list of agents (CTAs and CIAs) in the system and informs the CFA of the status (e.g., newly initialized) and information of these agents.

Aside from these functions, the SA is also responsible for all low-level interfaces. These includes access to the operating system or networking services, enforces access rights and privilege security, backs-up and archives pertinent information, and performs exception handling [1].

4 CONCLUSION

This paper presented the architecture of the CBR-TUTOR, an Internet agent-based tutoring system that uses the case-based reasoning approach in tutoring its learners. The architecture is designed such that it can be implemented for different domains (i.e., programming, problem solving, and others) and can be accessed through the Internet. The architecture differs from other internet-based tutoring systems because it utilizes the advantages of using previously experience cases to enhance the tutoring capability of the system. In addition, the CBR-TUTOR architecture is composed of specialized agents that performs the tutoring of learners, facilitation of requests, and filtering, retrieving and learning of new cases. All of these agents cooperate to achieve the goal of providing individualized and adaptive instruction to the learners. The use of agents in the design of this system is increases the effectivity of the tutor to provide adaptive instruction to its learners. Each of the tutoring agent focuses on the individual needs of its learners. Since the system is Internet-based, the use of agents can accommodate more users compared to non-agent-based systems. The other components of the system were also designed as agents because each of these components is autonomous and requires decision-making capability. Future work will focus on the implementation of

CBR-TUTOR in the domain of programming. Further research regarding the learning of cases where the system has the capability to do situation assessment where the reasoner elaborates a situation description to make the description fit the other case library descriptions will also be done.

References

[1] Abadia, L. (1999). A MultiAgent-Based Information Trading Floor Model for an Organizational Intranet.Graduate Thesis. Master of Science in Computer Science, College of Computer Studies, De La Salle University, Manila.

[2] Aamodt, A., Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7:1, pp. 39-59

[3] Chappell, A. and Mitchell, C. (1997). The Case-Based Intelligent Tutoring System: An Architecture for Developing and Maintaining Operators Expertise. Proceedings of the 1997 IEEE International Conference on System, Man, and Cybernetics. Orlando, Florida. http://www.chmsr.gatech.edu/publications/smc97/arc/abstract.html. (January 2000).

[4] Dingsoyr, T. (1998) Integration of Data Mining and Case-Based Reasoning. http:/www.idi.ntmu.no/ ~dingsoyr/ diploma/report.html. February 29, 2000.

[5] Kolodner, J. (1993). Case-Based Reasoning. Morgan Kaufmann Publishers, Inc.

[6] Reyes, R. (1999). Adaptive Web-Based Intelligent Tutoring System for C Programming. Proceeding of the 1999 International Conference in Computers in Education. Chiba, Japan.

[7] Row, R. (1995). A Case-based Reasoning Approach to Simulation-Based Intelligent Tutoring Systems and ITS Authoring. http:// www.navysbir.brtc.com/ SuccessStories/StottlerP3.html. (January 2000)

[8] Schank, R., and Osgood, R. 1990. A content theory of memory indexing. Northwester University, Institute for the Learning Sciences Technical Report no. 2.

[9] Sison, R . (2000). Multistrategy Discovery and Detection of Novice Programmer Errors. Machine Learning, 38, 157-180. Kluwer Academic Publishers, Netherlands.

[10] Weber, G. and Specht, M. (1997). User Modeling and Adaptive Navigation Support in WWW-based Tutoring Systems. ELM-ART II. http://www.psychologie.unitrier.de:8000/projects/ELM/Papers/UM97-WEBER.html. (January 2000)

[11] Wood, T. Psychological Access and the Internet (199). Advanced Research in Computers and Communications in Education. G. Cummings et al. (eds.). IOS Press. Pp.973-980.

Controlling Problem Progression in Adaptive Testing

Roger E. Cooley & Sophiana Chua Abdullah
University of Kent at Canterbury
Computing Laboratory, University of Kent at Canterbury,
Canterbury, Kent CT2 7NF, The United Kingdom
Tel: +44-1227-823816
Fax: +44-1227-762811
Email: rec@ukc.ac.uk, sc34@ukc.ac.uk

Adaptive testing has, in recent years, been used as a student modelling technique in intelligent tutoring systems. One of the main issues has been to optimise the progression of problems posed as the student performs the adaptive test. Previous research has concentrated on finding a structure in a fixed collection of problems. This paper describes an algorithm for problem progression in adaptive testing. After describing current approaches to the progression problem, the paper discusses the role of expert emulation. It then describes a knowledge elicitation exercise, which resulted in a solution to the progression problem. Part of the knowledge elicitation process was supported by software based on constraint logic programming, clp(FD), and the paper concludes with an assessment of the prospects of developing an extended knowledge elicitation support system.

Keywords: **intelligent tutoring system, knowledge construction and navigation, adaptive testing, constraint logic programming**

1 Introduction

The major advantages of adaptive testing over fixed item testing are that a student's knowledge is explored thoroughly and efficiently, and with a minimum of redundancy. By asking an appropriate number of problems at appropriate levels of difficulty, adaptive testing neither bores by unnecessary repetition nor intimidates by posing a series of inappropriately difficult problems [1]. This makes adaptive testing attractive for student modelling in intelligent tutoring systems [2],[3].

This research was conducted in the context of providing remedial help in mathematics to a transient population of prisoners in a local prison. Here the students are studying courses such as City and Guilds (Key Skills), City and Guilds (Number Power) and for GCSE level examinations. Working with prisoners can face tutors with problems not normally encountered in more conventional settings. Unlike school students, the prisoners not only lack uniform prior knowledge in mathematics, but tend also to join or leave the prison at individual times. This makes the job of the human tutor difficult because of the need to assess the knowledge level of each prisoner before assigning them the appropriate level of one or more of the above courses and examination. Currently, fixed item testing is used as an assessment tool. This approach has a major disadvantage. Many prisoners are ' math anxious' and the use of fixed item testing may undermine their confidence and motivation in the subject. Adaptive testing avoids this danger by presenting problems at an appropriate level of difficulty.

One of the main issues in adaptive testing is the determination of an efficient progression from one problem to another. Previous proposals have included hard-wiring prerequisite relationships between knowledge items [3], and preparing an indexing framework for problems [4]. Section 2 of this paper reviews the major lines of research; and the paper then describes an approach to the progression problem based on the knowledge acquisition techniques used for expert systems. In doing so, it continues in the vein of Khuwaja & Patel' s work [5]. The paper presents a rationale for this approach, describes briefly a semi-automated method of eliciting syllabus content and characteristics, and then presents a progression technique elicited by standard techniques with an expert. It concludes with a discussion of the feasibility of automation in this area.

2 The Progression Problem

In a problem-solving environment, *problem progression* is concerned with the strategy in which the next problem is selected. In adaptive testing, this is usually based on the student' s response to the current problem, as the process of selecting the next appropriate problem is crucial to the efficiency and precision of the whole student modelling process. Also, presenting *the right question at the right time* maintains the motivation of the student.

The structure of the domain, that is the way in which problems are related to one another, determines problem progression in adaptive testing; and the two significant and distinctive approaches to determining such structures are discussed in this section.

2.1 Item Response Theory

For adaptive testing systems which adopt the *Item Response Theory* or IRT [6], such as SIETTE [7] and CBAT-2 [8], the domain is made up of test items which are kept in an *item pool*. The construction of an item pool usually involves major empirical studies for content-balancing, to ensure no content area is over-tested or under-tested, and for item calibration. Each test item is associated with one or more of the following parameters – the *difficulty level, the discriminatory power and the guessing factor.* The *difficulty level* measures the difficulty level of a test item, the *discrimination power* describes how well the test item discriminates students of different proficiency, while the *guessing factor* is the probability that a student can answer the test item correctly by guessing.

Problem progression takes place like this. The adaptive test starts with an initial estimation of the student' s proficiency, è. A *best* item or problem is selected. This is one which provides the most information about the student, and is calculated from the item' s three parameters and current proficiency, è. An ideal item should have a difficulty level close to è, a high discriminatory power and a low guessing factor. A new proficiency, è' , and its confidence level are calculated based on whether the student has answered the problem correctly or not, the old è, and the item parameters. The test continues until a stopping criterion is met, for example, when the confidence level of è' has reached a desired level.

2.2 Knowledge Space Theory

There are adaptive testing systems built on the theory of *knowledge spaces[9]*. Examples include a web-based, domain-independent system called RATH [10], a web-based system for the domain of mathematics called ALEKS [11], and a general purpose system for testing and training called ADASTRA [12].

Like the IRT-based systems, the domain is made up of test items of an academic discipline, each of which can be a problem or an equivalence class of problems that the student has to answer. The student' s *knowledge state* is defined as the set of items in the domain that the student is capable of solving. For example, if a student has the knowledge state *{a,b,d}*, this means that he can solve items ***a***, ***b*** and ***d***. Not all possible subsets of the domain are feasible knowledge states. Consider the example shown in [13]. In a domain of mathematics, if a student can solve a percentage problem, (item ***d*** say), then it can be inferred that the student can perform single-digit multiplication, (item ***a*** say), and thus any state that contains item ***d*** would also contain item ***a***. The collection of all feasible knowledge states is called the *knowledge structure*. The knowledge structure must also contain the *null state {}*, which corresponds to the student who cannot solve any item, and the *domain*, which corresponds to the student who can solve or master all items. When two subset of items are knowledge states in a knowledge structure, then their union is also a state. This means that the collection of states is *closed under union*. When a knowledge structure satisfy this condition, it is known as a *knowledge space*.

In practice, items for a domain are derived from instructional materials and systematic knowledge elicitation with teachers. This is also the case with establishing knowledge states where query procedures systematically elicit from human experts the prerequisite relationships between items [3], [14].

Once the domain is represented as a knowledge space, the adaptive testing strategy is then to locate as efficiently and as accurately as possible, a student's knowledge state. Problem progression becomes straightforward. For example, if a student has answered an item correctly (incorrectly), it can be inferred that he can (cannot) answer a prerequisite item and will thus not be asked to solve the latter.

2.3 Other Approaches

The domain can be represented as a *granularity hierarchy* [15] where items which represent a topic, subtopic or skill, are described at various grain sizes and connected together into a granularity hierarchy which allows focus shifts along either *aggregation* or *abstraction* dimensions. In this way, the ability to recognise student behaviour at varying grain sizes is important both for pedagogical and diagnostic reasons.

Other examples include an *indexing framework* for the adaptive arrangement of problems in the domain of mechanics [4], a problem-simplification approach [16], an optimisation expert system where both the knowledge structures of the student and the teacher are represented by structural graph, and problem progression is controlled by the relationship between the student's knowledge structure and that of the teacher's [17]. Evidence of a strong use of a student model in controlling problem progression can also be found in a system called TraumaCASE [18] which automatically generated clinical exercises of varying difficulty, and in the work of Beck, Stern & Woolf [19] who recorded information about a student using two factors – *acquisition* and *retention*. *Acquisition* records how well students learn new topics while *retention* measures how well a student remembers the material over time.

3 Knowledge Elicitation

The concern of the researchers discussed above is to exploit a structure of a syllabus to improve the efficiency of tests. The structure may either be revealed through elicitation, as was done by Dowling and her co-workers, or may be derived from a statistical analysis of student behaviour, (IRT), or it may be seen as being derived from the nature of the problem domain. Though there may be, from some given point of view, an optimal way of structuring a syllabus, the view adopted in this research is that it is a subjective matter to be determined by an expert teacher. Such a teacher might make use of informal statistical information, subject domain information as well as pedagogic information in determining a suitable structure. Studies of intelligent tutoring systems have shown that, as one would expect, it is difficult to transfer systems from one setting to another, because there is considerable cultural variation in both teaching and learning [20]. This provides the prime motive for investigating techniques based on expert emulation for the production of tests for local consumption.

Moreover, this is a natural extension of the *intelligent tutoring systems* endeavour, and it has an additional advantage. A lack of homogeneity amongst a student body can weaken the effectiveness of techniques based on population statistics; and the target body of students with which this paper has been concerned is, educationally, not very homogeneous.

4 Eliciting the Syllabus

There are several problems to be confronted when adopting an expert emulation approach to designing an adaptive test. They include the problems of finding suitable experts [21], selecting appropriate forms of knowledge representation and choosing appropriate methods of knowledge acquisition.

The approach to knowledge acquisition in the research described here is to separate the task of designing an adaptive test into the following sub-tasks:

- describing classes of problems,
- describing the skills used to solve problems,
- describing responses to problems,
- problem generation,
- problem progression based on student responses.

For the particular domain tackled, namely the arithmetic of elementary fraction addition, software has been developed to support the first four of these subtasks using Constraint Logic Programming, clp(FD), embedded in Prolog, [22]. This work has been described in a recent conference paper [23], and is briefly summarised here.

Clp(FD) is actively used by the knowledge engineer conducting knowledge acquisition interviews. The teacher, who is the target of the emulation, is not expected to write constraints, but is more than likely to take an interest in them. During discussions, which involve the production of example problems, the knowledge engineer enters the necessary constraints, or modifies existing constraints, to describe the particular class of problem under discussion. The set of constraints is then solved interactively to produce example problems. These form the basis of a discussion, and may lead to further rounds of discussion and modification.

The description of a class of problems is treated as a set of constraints. This consists of a set of variables, a statement of the domains of the variables, and a statement of the relational constraints that hold between the variables. For example, during an interview, the human tutor wanted to represent a class of problems, which involved the addition of two proper fractions with a common denominator of the form,

$$N1/D1 + N2/D2 = N/D$$

and he wanted to use single-digit integers.

This can be represented in clp(FD) as a code fragment:

```
domain([N1,D1,N2,D2],1,9),      % Single digit integers
N1 #<  D1,                      % First operand - proper fraction
N2 #<  D2,                      % Second operand - proper fraction
D1 #= D2.                       % A common denominator
```

The following is an example of the use of clp(FD) to describe skills. The *cancel fraction* skill can be represented in clp(FD) as:

```
% Simplify the fraction N/D into its lowest form to give X/Y
% Example:  63/81 gives 7/9
cancel(N,D,X,Y) :-
      domain([N,D,X,Y,F], 1,99),
      F*X #= N,
      F*Y #= D,
      maximize(labeling([], [F,X,Y]), F).
```

Here, variable F is the common factor to be cancelled. This is specified by the two relational constraints. The *maximize* predicate in the final line ensures that the largest value of F will be found.

5 Eliciting the Progression

The knowledge elicitation exercise involved approximately 20 hours of interviews spread over a period of three months. Conventional knowledge elicitation techniques, such as structured interviewing, task analysis and construct theory [24], were used.

Early interviews revealed the significance to the expert of the *skills* that students needed to exercise in order to solve particular problems. The following were identified:

a. Add equivalent fractions
b. Cancel fraction
c. Make proper
d. Find the lowest common multiple
e. Find equivalent fractions

The number of discrete skills required to solve a problem was considered as a measure of the difficulty of the problem; and this measure was used to classify problems, and in so doing reveal a structure of the domain. This coincides with the findings of Beck, Stern & Woolf [19]. However, it is useful to note that this is only one of the many factors in measuring problem difficulty used by Lee [25], who identified, amongst others, the student's degree of familiarity with a particular type of problem.

In eliciting progression information, it is necessary to avoid the problem of combinatorial explosion. A *head on* approach requires the expert to provide a tree structure of sequences of problems indicating the appropriate *next* problem depending on the outcome of all previously asked problems. Such an approach is unattractive to both expert and knowledge engineer. Instead, an approach adopted was to attempt to uncover the underlying *algorithmic* strategy of the expert.

In general terms, the strategy of the expert is to test the students' abilities to exercise the identified skills at a particular level of difficulty. Failure to return a correct answer causes the questioning process to be resumed at a lower level of difficulty, that is, with problems requiring the demonstration of fewer skills. Whereas successful demonstration of all the identified skills causes the questioning process to be resumed with problems at a greater level of difficulty. The expert started with problems of middling difficulty and adopted a *binary chop* approach to selecting the next level. Within each level of difficulty, the selection of the *next* problem depended on the skills already demonstrated. Each available problem was scored using a set of weights, which favoured previously undemonstrated skills at that level. If the progression problem is viewed as a variant of state-space search, the expert's strategy has more in common with a *constrain-and-generate* paradigm [26], at a given level of difficulty, rather than a naïve *generate and test* approach. A schematic example of the use of this strategy is given below.

In a Prolog implementation of this strategy, a record of students' skills, demonstrated at each tested level of difficulty is recorded, and used to prepare a revision plan.

6 An Example

The human tutor first prepared the adaptive testing strategy for a domain of five skills described above. This is shown in Figure 1 for a domain of five skills.

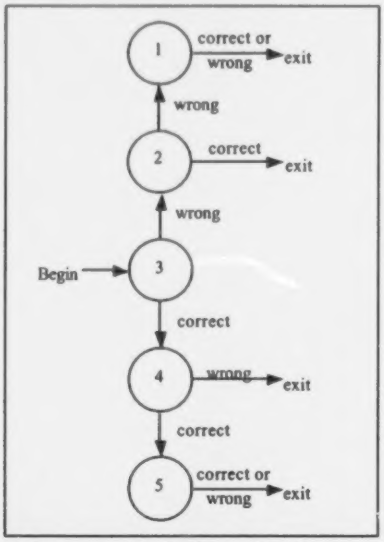


Figure 1: Human tutor's strategy in adaptive testing for a domain of 5 skills

In Figure 1, the adaptive test begins at node 3 which contains problems each of which can be solved by exactly three skills. If the student gets any problems wrong within that category, he moves onto node 2 which contains problems each of which can be solved by exactly two skills. If he gets all the problems correct within that category, he will exit the adaptive test. The rationalisation for this is described below.

If each of the skills were labelled as a,b,c,d,e, as in Section 5, then at node 3, there are ${}^5C_3$ that is 10 possible combinations of skills. For example, the combination [a,b,c] would involve a set of problems which each require all the skills **a**, **b** and **c** to be used. Skills **a**, **b** and **c** correspond to *add equivalent fractions*, *cancel fraction*, and, *make proper* respectively. However in practice, not all these combinations will be found in a valid problem type.

We introduced weights to each combination to enable the choice of the next best combination. We also imposed the following criteria for calculating the weight of each candidate set:

- If a skill has been not been asked yet, it carries a weight of 2
- If a skill has already been asked once, it carries a weight of 1
- If a skill has been asked more than once, it carries no weight
- Select the first set amongst the candidate set with the highest score

The following process shows how problems, each of which, require a combination of three skills are presented to the student.

a. Select [a,b,c] and scores are assigned to the other combinations, based on the above rules:

	[a,b,c]	[a,b,d]	[a,b,e]	[a,c,d]	[a,c,e]	[a,d,e]	[b,c,d]	[b,c,e]	[b,d,e]	[c,d,e]
1	•	4	4	4	4	5	4	4	5	5

b. Based on these weights, combination [a,d,e] becomes the next best choice and is thus chosen. The scores for the remaining combinations are recalculated.

	[a,b,c]	[a,b,d]	[a,b,e]	[a,c,d]	[a,c,e]	[a,d,e]	[b,c,d]	[b,c,e]	[b,d,e]	[c,d,e]
1	•	4	4	4	4	5	4	4	5	5
2	-	2	2	2	2	•	3	3	3	3

c. Combination [b,c,d] becomes the next best choice and is thus chosen.

	[a,b,c]	[a,b,d]	[a,b,e]	[a,c,d]	[a,c,e]	[a,d,e]	[b,c,d]	[b,c,e]	[b,d,e]	[c,d,e]
1	•	4	4	4	4	5	4	4	5	5
2	-	2	2	2	2	•	3	3	3	3
3	-	0	1	0	1	-	•	1	1	1

d. Combination [a,b,e] becomes the next best choice and is thus chosen.

	[a,b,c]	[a,b,d]	[a,b,e]	[a,c,d]	[a,c,e]	[a,d,e]	[b,c,d]	[b,c,e]	[b,d,e]	[c,d,e]
1	•	4	4	4	4	5	4	4	5	5
2	-	2	2	2	2	•	3	3	3	3
3	-	0	1	0	1	-	•	1	1	1
3	-	0	•	0	0	0	0	0	0	0

e. As there are no more candidate sets, no more problems are presented.

The above example shows that out of the ten combinations, only problems of combinations [a,b,c], [a,d,e], [b,c,d] and [a,b,e] were chosen. As described previously, the human tutor would consider the student' s previous performance and if any answers to problems were found to be wrong, he would assign problems at node 2 (see Figure 1). Conversely, if all the answers were found to be correct, he would assign problems at node 4 which require problems to be solved with exactly four skills.

The human tutor took the view that if a student has already tackled problems of three skills, whether he got them right or not, information gathered in packets of three skills need not necessarily apply to problems involving two skills. He considered that students may become anxious about problems which require more skills, and although some of the skills may well have been demonstrated in *easier* problems, the student may find it difficult to apply them in *harder* problems.

7 Conclusion

The paper describes the development of an adaptive test in the domain of elementary arithmetic, which required two styles of knowledge acquisition. The first is concerned with describing problems and skills, and it is computer-assisted; whereas the second is entirely manual and is concerned with the ordering, or progression, of problems to be posed to the subject of a test. However, based on this experience, work is currently underway to develop software to aid with eliciting details of progression. A valuable insight gained is that some degree of formalisation of the problem, as well as being convenient for the knowledge engineer, is also acceptable to the expert who helped with this work.

A possible significant difference between the research reported here and the work reviewed in Section 2 is that the approach to progression is not restricted to a fixed collection of problems. In view of Lee' s findings [25], it would be inappropriate to enforce the equating of difficulty with the number of skills. Evidence encountered during the knowledge acquisition experience suggests that the sheer clerical complexity of mapping out sequences of problems, lead to some draconian simplification on the part of the expert. The task ahead, is to find an appropriate balance between convenience and efficiency.

References

[1] Vispoel, W. P., Rocklin, T. R., & Wang, T., "Individual differences and test administration procedures: a comparison of fixed-item, computerized-adaptive, and self-adapted testing", Applied Measurement in Education, 7, pp. 53-79, (1994).

[2] Collins, J. A., Greer, J. E., & Huang, S. X., "Adaptive assessment using granularity hierarchies and Bayesian nets", Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), Montréal, Canada, June 12-14, pp. 569-577, (1996).

[3] Dowling, C.E. & Kaluscha, R., "Prerequisite relationships for the adaptive assessment of knowledge", Proceedings of the Seventh World Conference on Artificial Intelligence in Education (AI-ED'95), Washington DC, USA, August 16-19, pp. 43-50, (1995).

[4] Hirashima, T., Niitsu, T., Hirose, K., Kashihara, A., & Toyoda, J., "An indexing framework for adaptive arrangement of mechanics problems for ITS", IEICE Transactions on Information and Systems, E77-D, pp. 19-26, (1994).

[5] Khuwaja, R. & Patel, V., "A model of tutoring: based on the behavior of effective human tutors", Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), Montréal, Canada, June 12-14, pp. 130-138, (1996).

[6] Wainer, H., "Computerized adaptive testing: a primer", NJ: Lawrence Erlbaum Associates, (1990).

[7] Rios, A., Millan, E., Trella, M., Perez-de-la-Cruz, J.L.,& Conejo, R., "Internet based evaluation system", Proceedings of the Ninth International Conference on Artificial Intelligence in Education (AI-ED'99), Le Mans, France, July 19-23, pp. 387-394, (1995).

[8] Huang, S. X., "A content-balanced adaptive testing algorithm for computer-based training systems", Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), Montréal, Canada, June 12-14, pp. 306-314, (1996).

[9] Doignon, J. P. & Falmagne, J. C., "Spaces for the assessment of knowledge", International Journal of Man-Machine Studies, 23, pp. 175-196, (1985).

[10] Hockemeyer, C. & Dietrich, A., "The adaptive tutoring RATH: a prototype", Proceedings of the International Workshop on Interactive Computer aided Learning (ICL'99), Villach, Austria, (1999).

[11] ALEKS 2000, http://www.aleks.com (accessed on August 15 2000).

[12] Dowling, C. E., Hockemeyer, C., & Ludwig, A. H., "Adaptive assessment and training using the neighbourhood of knowledge states", Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), Montréal, Canada, June 12-14, pp. 578-587, (1996).

[13] Villano, M., "Probabilistic student models - Bayesian belief networks and knowledge space theory", Proceedings of the Second International Conference on Intelligent Tutoring Systems (ITS'92), Montréal, Canada, June 10-12, pp. 491-498, (1992).

[14] Kambouri, M., Koppen, M., Villano, M., & Falmagne, J. C., "Knowledge assessment - tapping human expertise by the query routine", International Journal of Human-Computer Studies, 40, pp. 119-151, (1994).

[15] McCalla, G., Greer, J., Barrie, B., & Pospisil, P., "Granularity hierarchies", Computers & Mathematics with Applications, 23, pp. 363-375, (1992).

[16] Hirashima, T., Kashihara, A., & Toyoda, J., "Toward a learning environment allowing learner-directed problem practice - helping problem-solving by using problem simplification", Proceedings of the Third International Conference on Intelligent Tutoring Systems (ITS'96), Montréal, Canada, June 12-14, pp. 466-314, (1996).

[17] Matsui, T., "Optimization method for selecting problems using the learner's model in intelligent adaptive instruction system", IEICE Transactions on Information and Systems, E80D, pp.196-205, (1997).

[18] Carberry, S. & Clarke, J. R., "Generating clinical exercises of varying difficulty", Proceedings of the Sixth International Conference on User Modeling (UM97), Ch ia Laguna, Sardinia, Italy, June 2-5, pp. 273-275, (1997).

[19] Beck, J., Stern, M., & Woolf, B. P., "Using the student model to control problem difficulty", Proceedings of the Sixth International Conference on User Modeling (UM97), Chia Laguna, Sardinia, Italy, June 2-5, pp. 277-289, (1997).

[20] Payne, S. J. & Squibb, H. R., "Algebra mal-rules and cognitive accounts of error", Cognitive Science, 14, pp. 445-481, (1990).

[21] Lightfoot, J. M., "Expert knowledge acquisition and the unwilling expert: a knowledge engineering perspective", Expert Systems, 16, pp. 141-147, (1999).

[22] Clp(FD) 2000, http://www.sics.se/isl/sicstus/docs/latest/html/sicstus.html#CLPFD (accessed on August 15 2000).

[23] Chua Abdullah, S. & Cooley, R. E., "Using constraints to develop and deliver adaptive tests", Proceedings of the Fourth International Conference on Computer Assisted Assessment (CAA), June 21-22, Loughborough University, UK, (2000).

[24] McGraw, K. L. & Harbison-Briggs, K., "Knowledge acquisition, principles and guidelines", NJ: Prentice-Hall, (1989).

[25] Lee, F. L., "Electronic Homework: an intelligent tutoring system in mathematics", PhD Thesis, The Chinese University of Hong Kong, (1996).

[26] Marriott, K. & Stuckey, P., "Programming with constraints: an introduction", Cambridge, MA: MIT Press, (1998).

Development of Intelligent Learning Support System with Large Knowledge Base

Katsuyuki SUGAYA and Setsuo OHSUGA
Department of Information and Computer Science, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169-8555, JAPAN
E-mail: {sugaya, ohsuga}@ohsuga.info.waseda.ac.jp

The objective of this paper is to present framework for developing intelligent learning support system with large knowledge base. Recently, the need for effective learning support and training is mounting, especially in industry or engineering fields, which demand the learning of complex tasks and expertise knowledge. Intelligent learning support system is being employed for this purpose, thus creating a need for cost-effective means of developing learning support systems. In this study, intelligent learning support system is assumed as a part of the intelligent knowledge management support system. The factors necessary for the intelligent learning support system discussed here are generality and adaptability. In order to achieve the goal, a new design of the system and learner modeling technique are discussed as well as a way of generating specific intelligent learning support system.

Keywords: **Intelligent System Design, Large Knowledge Base, Learner Model, Model-based Diagnosis, Knowledge Management**

1 Introduction

The purpose of this paper is to introduce a new framework for developing intelligent learning support system using large knowledge base. This system is a part of the intelligent systems that is developing to enable the expertise knowledge management.

In daily life, human has to interact with and reason about a large number of systems. This includes physical devices as well as non-physical systems. Also in professional work a growing number of people has to be trained in operating and designing large complex systems such as airplanes, nuclear power plants, and enterprises. Consequently, the goal of education or teaching may vary from inducing insight in the physical principles underlying the behavior of the device to teaching behavior analysis in the context of system design, operation, and maintenance. In addition, recently systems in the real world are becoming larger and more complicated. Rapid progress in science and technology has created a need for people who can solve complex problems and operate and maintain sophisticated equipment. In these situations, we, human beings, have to solve various types of problems using expertise in the large and complicated systems. Therefore the need for effective learning support or training is rising, given the increasing complexity of the workspace, especially in engineering or industrial fields.

Many computer assisted instruction techniques exist that can present instruction, and interact with students in a tutor-like fashion, individually, or in small groups [3]. The introduction of artificial intelligence technology and expert systems technology to computer assisted instruction systems gave rise to intelligent tutoring systems. In the intelligent tutoring system, for example, intelligent tutors that can model the learner's understanding of a topic and adapt the instruction accordingly [2]. Although intelligent tutoring systems research has been carried out for over 15 years, few tutoring systems have made the transition to the commercial market. Authors consider that some serious problems exist in the current methodology of developing intelligent tutoring systems. As an example, each system is developed independently, and tutoring expertise is hard-coded into individual systems. In particular, the problem of learner modeling technique exists as a basic issue. The system must have learner model that represents an estimate of the learner current understanding of the domain knowledge to be used by tutor in order to give adaptive guidance and explanations to the learner. A number of learner modeling techniques have been developed [8]. However, not every model can be called complete expressing the learning condition of the learner. Hence, the motivation for this study comes from the need for effective intelligent tutoring systems, particularly development of more complete learner modeling technique.

For these problems like above we consider that the factors necessary for the intelligent learning support system discussed here are generality and adaptability. In order to achieve the goal, authors present a new framework of the intelligent learning support system those enough practical conditions. Several concepts are included in this study; expert knowledge management with large knowledge base, knowledge sharing, knowledge processing, model-based learner diagnosis, etc.

2 Expert Knowledge Management using Knowledge Base System

In this section, we introduce briefly the key concept of our knowledge base system. Our research groups have tried to solve various problems by knowledge-centered intelligent system. The main concept is Multi-strata modelling scheme [5]. This modelling scheme is applied many intelligent systems, and these systems rewarded with good results, e.g. automatic programming system [1]. And we considered that Multi-strata model is strongly support the development of intelligent tutoring systems [6][7].

2.1 Intelligent System with Large Knowledge Base

At first, we discuss to apply large knowledge base for the architecture of intelligent learning support systems, which can generate learning support systems for a wide range of domain.

In these days, with the developing of science and technology, the systems which human manages with are enlarged and more complicated. In particular, it is too difficult to transmit expert knowledge from expert engineer to novice engineers. In the engineering field, even a large system developed by many expert engineers. When the system grows lager and more complex, the knowledge that is needed to build the system is more specialized and subdivided. In these situations, some serious problems are occurred. For instance, it is difficult to communicate between expert engineer and another fields' engineers or novice one. In other words, it is too more expertise to transmission of expert knowledge from human to humans. For this reason, the expert knowledge hiding is occurred in some engineering companies.

When the knowledge is specified and subdivided, in the situation like classroom, it is not appropriate to transmit the knowledge from expert engineer to novice one e.g. next generation engineers. Therefore, we propose the transmission of expert knowledge through the large knowledge base system (Fig.1).

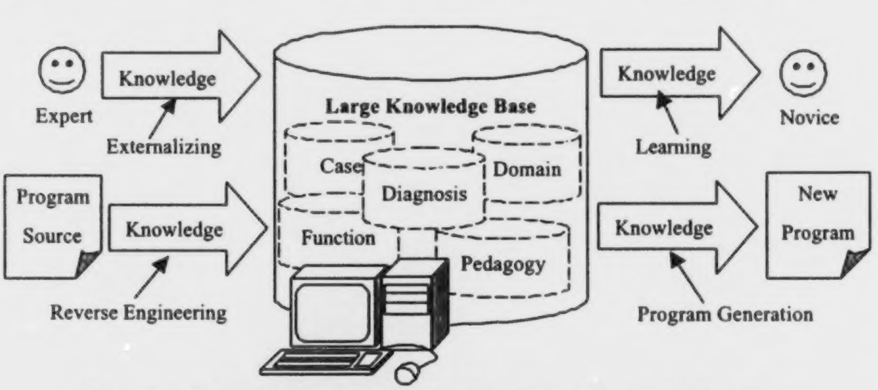


Fig.1: Knowledge Transmission using Intelligent Large Knowledge Base System

In this study, we consider that intelligent learning support system is a part of intelligent knowledge management support system. Moreover, we believe that knowledge management or learning support system is one of large and complex problem solving systems. The term problem is used here in a wide sense to mean what a person wishes to know or wants to do. There are various types of problems such as analysis, design control, decision-making, planning, and teaching. Most of them are not well dealt with by conventional software method but require the system a capability to find a solution itself in a large space. Since the space is open, self-controlled exploration in the space is necessary. The system must be provided with the various methods to solve the different type of problems, each of which is represented by a specific knowledge chunk. Furthermore, a complex problem concerns different problem domains and since a problem requires domain specific knowledge, the system must be provided with a global knowledge base including the various type of domain knowledge.

In order to use knowledge effectively, the system must be able to extract only the necessary knowledge from the knowledge base referring to the type and the domain of the problem to be solved. For this purpose knowledge must be well structured. All used knowledge is accumulated in the large knowledge base (Fig. 1).

2.2 Necessity of Knowledge Processing Language

The whole of the problem solving process is from accepting external representation of problems to generating solutions. In order to represent problems in the system a processing language is necessary. The language has to meet two conditions: it has to be usable for representing problems; and it has to be processable by computer processor. In ordinary computers only the procedural language is used both for processing by the processor and for representing problems. The knowledge base system, on the other hand, introduces the second language to separate the above two aspects, as well as a conversion mechanism between them. The second language is a declarative knowledge representation language. The conversion either in the declarative forms or from the declarative to the procedural form is necessary. This is the inference. It can be implemented as a procedural program on conventional computers.

The specification for the second language must be decided so that it can represent these conditions. It had to be suited for representing predicate including data structure as argument and also for describing higher-level operation such as knowledge for selecting object knowledge. KAUS (Knowledge Acquisition and Utilization System) has been developed for the purpose by our research & development team.

3 Adaptability of Learning Support System

To meet the condition of adaptability, it is necessity to represent the learner's understanding of learning domain. In this section, we discuss a learner modeling method that is applied to diagnostic techniques in artificial intelligence.

3.1 Issues of Learner Model

The performance of intelligent learning support system depends largely on how well it knows why the learner fails to solve problems. Because of the sophisticated interaction requires information about the learner, the system has to maintain some kind of model of the learner. This model may include cases about what has been done before or information about what the learner is believed to know. The process of gathering information about the learner is mostly referred to as cognitive diagnosis. Ohlsson has given a widely accepted definition of cognitive diagnosis: "cognitive diagnosis is the process of inferring a person's cognitive state from his or her performance" [4]. We consider that the point of learner model is to represent knowledge state of learner, especially his/her fails to solve problem. To satisfy this requirement, we focus diagnosis techniques.

A diagnosis is defined in terms of one or more reasoning steps that the learner cannot have solved problem. A major advantage of this approach is that it can be based solely on a model of these correct reasoning steps; no knowledge is required about the specific misconceptions that learners may have about the domain of learning. Instead we model all primitive inferences that are required to arrive at the correct solution. In addition, our approach to diagnosis of learner behavior exploits results from model-based diagnosis as it is defined in the field of artificial intelligence.

3.2 Model-based Learner Diagnosis with Case Base

Model-based diagnosis is a prominent area within artificial intelligence and emerged in the last about 15 years. The technique of model-based reasoning has been widely researched and accepted as the principal diagnosis in electronic circuit analysis, power station maintenance, medical diagnosis domains, etc. However, little emphasis has been put on its application to education or training system domain. The basic principle in model-based diagnosis is the description of system as a causal model. With the model at hand, the behavior predicated by the model is compared to the actually observed behavior. Since the predictions of the model are based on the assumption that the components work correctly, these assumptions may be partially dropped to accommodate for a detected behavior difference and thus diagnose faulty behavior.

However, there are some weaknesses in model-based diagnostic technique. The most serious weak point is the diagnosis time. It sometimes takes so much time to diagnosis. Therefore, we must be considering that it is necessary to model concerning the trade-off between the cost of a diagnosis time and its precision. Case-based reasoning, by contrast, excels in covering weak-theory domains, domains whose phenomena we do not yet understand well enough to record causality unambiguously. This feature allows case-based reasoning to be used in domains where model-based reasoning cannot be applied.

In the case-based reasoning, a reasoning engine remembers previous situations similar to the current one and uses them to help solve the new problem. However, case-based diagnostic technique has been criticized on many grounds. For example, that being specific to the system being diagnosed, they are non-constructive and that, having no analytic basis, the methods are restricted to specified faults and have a known level of competence. We think that the model-based diagnosis, being independent of the particular device descriptions, is intended to overcome these difficulties.

Therefore, we consider developing the approach of the model-based diagnosis system with case base. Model-based reasoning and case-based reasoning have the potential to complement each other quite well. However, no work has been done on specific issues of learner modeling using combine model-based reasoning with case-base. The outline of model-based learner diagnosis with case base is following. When the set of learner's behavior data input the diagnosis system, the diagnosis engine reasons the state of his/her knowledge consulting the diagnosis knowledge base include case base and object model base. The design of the model-based diagnosis system begins from describing the system as diagnosis object model. The system, which is a diagnosis object, is considered to be a set of domain models. The diagnosis object model that has knowledge of proper action, and the set of the behavior of learner as input value are given to a system. The first behavior of the system that received input is to seek whether there is a history about the same case in the case base. If the record to apply in the case is found, case base returns list of learner's knowledge, which should examine to diagnosis engine. Diagnosis engine does investigation about domain model of each record given to it, by comparing a simulation result in object model with the actual behavior of learner. Diagnosis process is finished if a trouble is recognized. When there was no record that complied with the input value in the case base, the process starts to use diagnosis domain object model. This domain object model has the hierarchical structure. A process begins from making the error model that one component in the extreme high class in the diagnosis object model is supposed to be out of order. The purpose of this process is to simulate using a made error model to examine whether the result of the simulation is the same as the behavior of learner. If there is no contradiction in the simulation result, the model-based reasoning is done again toward each domain knowledge model of the lower layer. In the same way, a diagnosis process is repeated until a trouble is recognized in knowledge component of the extreme lower layer. All process of diagnosis is knowledge processing by KAUS.

4 Discussion

The objective of this study was to develop a new intelligent learning support system, especially to focus two conditions; generality and adaptability. Authors in first propose the architecture of intelligent learning support system with large knowledge base to enough generality, which modeled by using multi-strata model. In second presented the model-based learner diagnosis to meet adaptability. All of the knowledge was represented by KAUS in intelligent learning support system that was assumed a part of the intelligent problem solving system. The issue of learner diagnosis is very important point to achieve adaptive instruction in intelligent learning support system. We proposed that fault diagnosis techniques be applied to infer the state of learner's knowledge. So we discussed the feature of diagnostic techniques, especially model-based reasoning with case base. Model-based reasoning appears to be a more promising technique than other knowledge-based methods because it can diagnose the faults that have not been pre-determined. Fails in learner's knowledge can be diagnosed automatically based on the models, which describe the correct behavior. However, because model-based approach reasons from the actual structure and function of knowledge, it is inefficient for some problems. Furthermore, obtaining domain models is sometimes either difficult or too complicated, whereas most of the fails can be diagnosed based on past experience, which is very effective if the rule base or the case base is either comparatively small or well-indexed. A better solution is a hybrid approach integrating some of the diagnostic approaches. A case base will be provided to access the solutions to some fails diagnoses occurred previously, of which the domain models are unavailable. For some diagnoses, their solutions and contexts can also be stored in the case base for reuse later. Frequently occurring fails can be diagnosed efficiently even by a few of heuristic diagnostic rules. We believe that such a hybrid diagnostic approach will perform better than any of them does. In order to achieve this goal; we have considered the division of object model and problem type. On this part, it is necessary to carry out examination that will be more profound in future work.

References

[1] Aida, T. & Ohsuga, S., "Intelligent Systems Must Be Able to Make Programs Automatically for Assuring the Practicality", Lecture Note in Artificial Intelligence 1415, pp.47-56, (1998).

[2] Costa, E, New Directions for Intelligent Tutoring Systems, Springer-Verlag Publishers, New York, (1991).

[3] Mandl, H. & Lesgold, A, (eds.), Learning Issues for Intelligent Tutoring Systems, Springer-Verlag Publishers, New York, (1988).

[4] Ohlsson, S., "Some principles of intelligent tutoring", Instructional Science, Vol. 14, pp.293-326, (1986).

[5] Ohsuga,S., "Multi-Strata Modeling to Automate Problem Solving Including Human Activity", Proc.Sixth European-Japanese Seminar on Information Modeling and Knowledge Bases, (1996).

[6] Sugaya,K. & Ohsuga,S., "Design of Intelligent Education System using Multi-Strata Modeling Concept", Proc. The Sixth International Conference on Computers in Education, Vol.1, pp.305-311, (1998).

[7] Sugaya,K. & Ohsuga,S., "Framework for Developing Intelligent Tutoring System Using Multi-Strata Model", Advanced Research in Computers and Communications in Education, Vo.1, IOS Press, pp.832-835, (1999).

[8] Wenger,E., Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge, Morgan Kaufmann Publishers, Los Altos, California, (1987).

Educational Agents and the Social Construction of Knowledge: some issues and implications

Carolyn Dowling
Australian Catholic University
115 Victoria Parade, Fitzroy, Victoria 3065, AUSTRALIA
Phone: (61 + 3) 9241 4456
Fax: (61 + 3) 9241 4546
Email: c.dowling@patrick.acu.edu.au

The use of intelligent software agents within computer mediated learning environments is currently an important focus of research and development in both AI and educational contexts. Roles envisaged and implemented include those of tutor, of 'manager', of information seeker and of fellow learner. Each of these raises its own special challenges in relation both to the capabilities of the software and to our understandings in regard to the nature of the learning process. High on the list of factors currently believed to contribute to effective learning is social interaction in the service of knowledge construction. Within many electronic learning environments we are currently witnessing the emergence of a new participant in the social interactions that mediate learning. The substitution of computer programs possessed of varying degrees of intelligence, autonomy and 'personality', for certain dimensions of human presence within the computer based classroom raises a number of questions related to the processes through which knowledge is socially constructed, and to the qualities which are necessary to ensure successful participation in those processes. Through discussion of both theoretical perspectives and practical examples, this paper explores some of these issues.

Keywords: **AI in Education, Educational Agents, Intelligent Tutoring Systems, Interactive Learning Environments, Networked Social Learning, Teaching and Learning Process**

1 Introduction

Developments in computing and information technology in recent years have rapidly propelled the notion of intelligent software agents from concept to implementation. Today, whether or not we are always aware of them, they are an integral part of a growing number of computing environments. From the invisible armies of knowbots and related entities scurrying around the Net in the service of increasingly sophisticated search engines to the cheery little characters who pop up on our screens offering assistance with anything from formatting a date to constructing a complex multimedia presentation, or the 'personalities' with whom we interact in chat rooms in happy ignorance of their purely digital nature, intelligent agents are alive and well and are multiplying rapidly.

An early but still useful conception of a software agent is, "A character, enacted by the computer, who acts on behalf of the user in a virtual environment", useful in mediating " ... a relationship between the labyrinthine precision of computers and the fuzzy complexity of man [10, p. 355]. Later definitions tend to be expressed in more functional terms, such as, "An agent can be viewed as an object which has a goal and autonomously solves problems through interaction, such as collaboration, competition, negotiation and so on" [9]. This definition has some similarities with that offered by Maes [12] who defines an agent as:

"A computational system which:

- is long lived;

- has goals, sensors and effectors;
- decides autonomously which actions to take in the current situation to maximize progress towards its (time-varying) goals" [12, slide 5].

Summarising the writings of a number of researchers, Aroyo and Kommers [1, p. 237] identify four major characteristics of agents as being autonomy, responsiveness or reactiveness, pro-activeness and social ability. Other qualities frequently proposed, but not supported by all researchers or indeed by all users, include the ability to learn from experience and consequently to respond in flexible and possibly unforeseen ways to particular situations, and the possession of a believable 'character' or personality as a basis for social interaction.

It appears that a combination of factors has contributed to the current proliferation of software agents. Apart from the technical developments which have opened up the possibility of implementing what were previously largely theoretical conceptions, there is our very real need for assistance as we operate within computing environments characterised by rapid change, large quantities of extraordinarily complex information, and a lack of common organisational structures through which information may be accessed and managed. As Laurel predicted, there are now many situations in which, in the interests of efficiency, some form of 'intelligent' mediation is required between computer systems and the needs of users.

There are, of course, different forms that this mediation could have taken. The strong propensity for most users to accept assistance in the form of a more or less personified entity as largely unproblematic undoubtedly derives at least in part from the anthropomorphic elements implicit in most computer interfaces from the earliest days of computing. It can be strongly argued that a degree of personification has always been automatically and inevitably conferred as much by a program's use of language as a component of the interface as by our everyday understandings of the 'intelligence', albeit artificial, of computers. Intelligence and language use are, after all, key defining attributes of human beings.

Not only are we accustomed to interacting with computers as though they share with us a degree of 'humanity', but in a number of areas of activity we have been persuaded to value 'social' interaction particularly highly. Education is a good example, given the extent to which our current understandings of learning depend upon an acceptance of the belief that knowledge is to a large extent socially constructed. In the current drive to move teaching and learning online, the notion of agency in computing has found a strong ally and a vehicle for expansion. Unless the social interactions that mediate learning in face to face environments can be shown to have a digital equivalent, proponents of online courses will be forever 'on the back foot', with their products being regarded by most educators as second best. While courses incorporating the communications facilities of the Internet certainly go a considerable way in promoting interactions of various types between teacher and student and also between student and student, the possibility of using software agents to create an illusion of interpersonal interaction so convincing as to achieve pedagogical outcomes equivalent to those deriving from a relationship with another human being is extremely enticing to the designers of electronic learning environments.

2 Some examples of socially interactive pedagogical agents

Johnson [7] has proposed the following definition the role of a pedagogical agent as distinct from those designed for other purposes:

> "Pedagogical agents are autonomous agents that support human learning, by interacting with students in the context of interactive learning environments. They extend and improve upon previous work on intelligent tutoring systems in a number of ways. They adapt their behaviour to the dynamic state of the learning environment, taking advantage of learning opportunities as they arise. They can support collaborative learning as well as individualized learning, because multiple students and agents can interact in a shared environment. Given a suitably rich user interface, pedagogical agents are capable of a wide spectrum of instructionally effective interactions with students, including multimodal dialog. Animated pedagogical agents can promote student motivation and engagement, and engender affective as well as cognitive responses" [7, p. 13].

This is a comprehensive and optimistic vision, incorporating a number of possible roles for software agents within educational environments. Types of agents currently implemented in projects around the world include record keepers, information seekers, testers, facilitators of collaboration, tutors or instructors, fellow learners, and tutees. Of special interest in regard to this paper are those that contribute to the overtly social dimensions of the learning environment. The last three listed most clearly fulfil this criterion.

2.1 Agents as instructors

There is a sense in which perceptions of the role of computers in the learning process have come full circle. Early models of the role of 'computer as tutor' in the form of drill and practice style of instructional software, generally based on Skinnerian principles and incorporating very limited interaction between user and computer, have long been rejected by most educators in favour of a range of other more acceptable guises including that of a learning tool, an information source, and a learning 'space'. With the development of agent technologies, as Johnson suggests, new possibilities now exist for incorporating computers within the learning environment in a range of socially interactive roles, including that of 'tutor', through modes of interaction more in keeping with current pedagogical theory.

It is commonly asserted that the presence of computers in classrooms has itself played a part in modifying the image of the teacher as the 'sage on the stage' in favour of a more collaborative model. Not surprisingly, these changing concepts are well reflected in many implementations of 'agent as teacher'. As Solomos and Avouris [18] write, for instance:

> "The user mental model of the system should be based on the metaphor of the "invited professor" rather than the "knowing everything own tutor". ... Our first findings confirm the observation that today's users, accustomed to hypertext-like interaction, are more likely to accept this collaborative teaching metaphor, according to which their tutoring system is viewed as an intelligent hypertext browser, offering links to other tutoring systems with the right content and at the right time" [18, p. 259].

The increasingly popular concept of the teacher as a facilitator of learning is also reflected in such statements as: "Each student working on the project will have an agent, operating in the background, watching progress, measuring it against the plan, and taking remedial action when necessary" [19, p. 362].

2.2 Agents as fellow learners

A style of agent of special significance in the context of socially constructivist theories of learning is the 'fellow learner', which to differing degrees might be presumed to include all participants within the learning environment. If agents are to gain widespread acceptance in the field of education, this is an important area for research and development. Since the 1980s Chan [2, 3] and colleagues have been working on a range of models of socially interactive agents for learning environments, perhaps the best known being the 'learning companion' – a software entity having limited knowledge of the domain in question, conceptualised as a fellow learner with whom the student may collaborate and even disagree. As in real life, some of these learning companions may be better informed than the student in the relevant domain of knowledge, while others may know less. Perhaps not surprisingly, in learning environments for younger students, animals are a popular choice of persona for such agents, as in this example of a networked learning environment for Taiwanese high school students, as described by Chan:

> "The Dalmation is having the same performance as the student. ... Another animal companion is Dragon, like one of those animal companions in Mulan, a Disney cartoon of this summer. This dragon will "learn" (mainly rote learning) from the student and also from other students on the Net and so may know more than the student. At certain point it'll stop learning and come back to teach the student. In a way, Dragon is protecting the student" [3].

An interesting development of this concept is presented by Sheremetov and Nunez [16, p. 310], who describe the function of a 'monitor agent' as being to modify the role, behaviour or expertise of learning companions from that of strong group leader to a weaker companion or even a passive observer, depending on its interpretation of the degree of guidance required by the learner.

2.3 Agents as pupils

We are all familiar with the common wisdom that we learn through teaching others. At the school level, many educators have long been familiar with the concept of the computer as 'tutee' through the use of the Logo programming language, in which 'teaching the turtle' was a familiar metaphor for the activity of programming. More recently, a number of researchers have explored the translation of this concept into electronic learning contexts where agents exist to be 'taught' by the student user, as in the example from Chan quoted above. A further example is described by Ju [8] who writes of a computer based peer tutoring system employing two categories of agent – an 'expert', and a 'learner':

> "… students become active learners who are guided to learn by teaching a computer. After the students watch how the computer expert solves a set of linear equations [the program] helps the human student act as a teacher in order to learn more about the subject matter. At this time, the computer plays the role of a student ..." [8, p. 559].

3 Some issues for consideration

3.1 Multiple agents

Most agent based systems utilise a number of agents, many of them capable of a complex range of interactions with the student, with one another, and increasingly with agents associated with other programs. Their individual purposes derive from theoretical analyses of the component tasks and activities that are included in the larger scale pedagogical interactions of human beings. As educators, and indeed as students, we may simultaneously enact a range of roles within the educational environment. The apparently unitary activity of 'teaching' involves such elements as demonstrating, guiding, telling, questioning, explaining, testing, motivating, criticising – even learning! Many researchers consider that the electronic medium makes it feasible to identify and separate out these diverse functions. These can then be enacted through different configurations of agents working in relationships which ranging from collaboration to competition.

An example is the Multiple Agent Tutoring System (MATS) described by Solomos and Avouris:

> " MATS is a prototype that models a "one student-many teachers" learning situation. Each MATS agent represents a tutor, capable of teaching a distinct subject. All MATS tutors are also capable of collaborating with each other for solving learning difficulties that their students may have" [18, p. 243].

Strategies for most efficaciously combining the activities of multiple agents such as these necessitate a complex agent architecture, and understandably occupy a great deal of the research agenda in this area. Of interest in relation to their participation in the social construction of knowledge is the fact that one of the most common metaphors employed by a number of researchers and courseware designers is that of a 'society' of agents, a conception reminiscent of Minsky's *The Society of Mind* [14], Gardner's multiple intelligences [6] and other related theories of cognition and behaviour. In describing the different aspects of the design of their "multi-agent, computer-based interactive environment", for example, Costa and Perkusich [4, p. 196], drawing on the work of Franklin and Graesser [5] refer to their aggregation of agents quite specifically as a 'society'.

> "The society [of artificial tutoring agents] is an open multi-agent system made up of a collection of tutoring agents that co-operate among themselves to promote the learning of a certain human learner. This society is designed to be open and dynamic in the sense that it allows maintenance operations such as the entry and the exit of agents, besides eventual modifications in the knowledge and in the inference mechanisms of an agent. Each agent defines an expert tutor in some domain, having the necessary knowledge to solve problems in this domain. These agents are cognitive and possess properties like autonomy, goal-oriented, social ability" [4, pp. 197-198].

While on the one hand, the variety of functions of agents within a multi-agent environment must also be appreciated as an attempt to realise the type of rich user interface which Johnson suggests is necessary if the pedagogical interactions within electronic learning environments are to approximate to any degree to the face to face educational experience, some educators have concerns in regard to the assumptions underlying these practices. They argue that such developments are underpinned by a reductionist rather than a holistic understanding of the processes and relationships involved in teaching and learning. In separating out the different components of pedagogical interactions, are we enabling each part to be realised more effectively, or are we failing to acknowledge that the global act of human teaching may in fact be more than the sum of its component parts? It seems reasonable to suggest that firm judgments on issues such as this must await greater experience of the roles of agents within these learning contexts.

3.2 Personification

Another focus of debate concerns the degree to which personification is helpful in fostering fruitful pedagogical interaction between the human learner and software agents. This question clearly relates more to the 'socially interactive' agents than to those fulfilling more tool-like functions, which arguably require far less in the way of 'personality'. As noted earlier in this paper, there are clear arguments for accepting that a degree of personification of computer interfaces is inevitable. As Shirk puts it:

> "Although there is some dispute among software critics concerning the advisability of having 'personalities' in computer programs, their presence seems unavoidable. Any time there is communication between a computer and a human, the information presented by the computer has a certain style, diction, and tone of voice which impact upon the human's attitude and response toward the software" [17, p. 320].

However the extent to which this should be deliberately fostered is less clear, although many feel intuitively that it should be an important element in the creation of an electronic learning environment characterised by interactions which can reasonably be described as 'social'.

An important aspect of the representation of 'character' or personality is visual appearance. Interestingly, both research and experience suggest that the relationship in the case of software agents is far from straightforward, and that a mismatch between realism in appearance and the apparent knowledge level of the agent can have a deleterious effect on credibility. The more visually realistic the representation, the higher the expectations of the user in relation to the appropriateness and 'intelligence' of utterances and actions. Agents that 'look' smart and 'act' or 'talk' dumb are poorly received by many users, who express a higher tolerance for the limitations of a 'character' more sketchily represented, for instance through cartoon-like graphics. As Masterton, writes, for instance, "A common problem with AI programs that interact with humans is that they must present themselves in a way that reflects their ability. Where there is a conflict between the ability of the system and the users' perception of that ability a breakdown occurs and users may either fail to exploit its full potential or become frustrated with its shortcomings" [13, p. 215]. He goes on to suggest the implementation of a degree of anthropomorphism intended to convey qualities such as friendliness and usefulness, without the implication of possession of full human capabilities [13, p. 211]. He describes the development and role of such an entity in the form of a VTA (Virtual Teaching Assistant) which is able to introduce topics and answer simple questions, the more complex types of exposition and interaction being left to the human teacher. In terms of a traditional scenario at university level, the VTA functions somewhat like a tutor or demonstrator as distinct from a lecturer. "In this way faculty is left free of the guiding and assisting issues of the course and is able to concentrate on more complex questions and higher level issues generated during the course" [13, p. 211].

Further instances of this principle are the examples of agents presented as animals discussed earlier in this paper. Our expectations in regard the cognitive skills of animals may well be more appropriate to the capabilities of software agents than are our experiences of human-to-human interactions.

3.3 Autonomy

Closely related to the 'intelligence' of software agents is the issue of autonomy, in particular the degree to which an agent should be furnished with pre-existing goals which might lead it to take particular action without instruction from the user, and even contrary to what the user might perceive as his or her interests and wishes. Exploring the implications of such entities existing and interacting within virtual reality environments, Loeffler [11], for instance, notes that the unpredictability resulting from significant autonomy might well result in agents who are less 'helpful' to us than we might hope or indeed expect. It is easy to slip from such considerations into the need for a contemporary version of Asimov's laws of robotics as conceived in fictional terms more than 30 years ago!

In educational contexts, the implications of autonomy, particularly in terms of control of and responsibility towards the learner, are potentially extremely complex and difficult to address without more exposure to these types of software, and indeed it is quite likely that such experience may cause community understandings in regard to appropriate relationships between the 'human' and the 'not human' in electronic contexts to develop and change over time. In the short term, current trends in educational thinking which favour giving more control and autonomy to the learner would appear to be more in line with the thinking of researchers such as Schneiderman who favour 'direct manipulation' over the development of interactive agents with a significant degree of independence of action. Where agents are involved, they may be programmed so as to exercise control over the learner on behalf of the creator of the learning environment, or they may be configured so as to be more sensitive to a user model, and more responsive to instruction from the user/student. In the latter instance, the agent would have a greater degree of responsibility to the needs and wishes of the learner, but this may not be in keeping with the pedagogical goals of the teacher.

Trust is another aspect of the teacher/learner relationship that is complicated by the degree of autonomy with which a pedagogical agent is endowed. To the extent that the programmer chooses to delegate certain functions and responsibilities to the agent, it is their problem, but it may also be an issue for students, particularly those with more insight into the nature of the agents with which they are interacting.

A further concern in regard to the autonomy of pedagogical agents relates to the issue of intervention in the learning process. Despite the finding of Aroyo and Commers [1] that pro-activity is a quality frequently sought after in agents, there is an important issue of balance to be addressed in relation to the educational process. It is well accepted that a high degree of unsought assistance whether from a human teacher or an excessively diligent and proactive agent can be quite detrimental, in particular to the metacognitive aspects of learning. Of course this is also an issue for teachers and learners in face to face educational contexts!

3.4 Level of participation in the social construction of knowledge

The belief that it is possible for agents to participate effectively in the social aspects of knowledge construction is central to the work of many theorists and researchers. Sheremetov and Nunez [16], for example, whose works derives overtly from the theoretical frameworks of Piaget and Vygotsky, argue that:
"The design of learning environments, virtual or not, aims to promote productive interactions. In this type of learning a student changes from being a passive information receiver to an active collaborator, interacting with the tutors and colleagues in the learning process. Learning does not only result from acquiring knowledge, solving problems or using tools, but also from interacting about these on-going activities with persons and agents"[16, p.305 – 306].

In relation to their specific project they write: "Our emphasis lies in the role of interactions in an artificial learning community as a group of real and artificial learners, tutors, and facilitators, working, supporting and learning from each other [16, p. 306]. But however personified and autonomous the software agent, can it really be said to participate fully in the social construction of knowledge? It has been argued quite extensively that even the most heavily personified of computer programs suffer from an intrinsic lack of ability to participate in the metacognitive aspects of learning. Pufall [15], for instance, expresses a strong belief that a computer program is unable at any level commensurate with human capacities to modify its own knowledge structures or cognitive processes, and so cannot be regarded as a co-constructor of knowledge in a meaningful sense. While this might well have been the case in relation to earlier computer based learning environments, can we continue to make the same claims with confidence today or in the future? The capacity of software to 'learn' and adapt to experience through the incorporation of new information, the appropriate modification of its representation of the context in which it functions (its 'world') and of its inference mechanisms, is undoubtedly increasing. One way of considering this question might be to look at it in terms of the type of distinction sometimes made between 'hard' and 'soft' notions of artificial intelligence. If our test of full participation depends on an understanding that the agent has 'learnt' in precisely the same way that the human has learnt, then we will have difficulty accepting the electronic entity as genuine co-constructor of knowledge. If, however, we make our claim on the grounds that it appears to the human learner that the agent has participated in the learning that has taken place, then perhaps we can at least tentatively admit such a piece of software to membership of the social milieu which has mediated the educational experience.

Conclusions

It is clear that developments in agent technology have created a range of new possibilities in terms of aligning computers more strongly with prevailing educational theories and philosophies. In considering the many issues which might be raised in relation to the nature and roles of pedagogical agents, there are three overarching questions. Firstly, do agents have the potential to enhance learning, or do they threaten to undermine those aspects of the educational enterprise that we most value? Secondly, to what extent might they assist in the replication of the social dimensions of face to face learning within online environments? Thirdly, do they go further than this, and create new possibilities in regard to the social mediation of learning? To the extent that visions such as those of Johnson [7] are able to be realised, we may be faced one day with the need to re-evaluate our attitudes regarding the relative merits of a human teacher and an electronic entity designed specifically for educational purposes. But while the rhetoric of developers often suggests an ideal surpassing the sometimes imperfect realities of human-to-human pedagogical interactions, the 'jury' of online learners and of educators is still out.

References

[1] Aroyo, L & Kommers, P., "Preface – Intelligent Agents for Educational Computer-Aided Systems", Journal of Interactive Learning Research, Vol. 10, No. 3/4, pp. 235 – 243, (1999).

[2] Chan, T.-W., "Learning companion systems, social learning systems, and the global social learning club", Journal of AI in Education 7 (2) (1996).

[3] Chan, T.-W. (1998), "The past, present, and future of educational agents", http://www.apc.src.ncu.edu.tw/apc/ppt_chan.html (accessed on February 3, 2000).

[4] Costa, E.deB. & Perkusich, A., "Designing a Multi-Agent Interactive Learning Environment", in Proceedings of International Conference on Computers in Education (ICCE '97), Kuching, Sarawak, Malaysia, December 2 – 6, pp. 196 – 203, (1997).

[5] Franklin, S. & Graesser, A., "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Berlin: Springer-Verlag (1996).

[6] Gardner, H., "Frames of mind: the theory of multiple intelligences", New York: Basic Books (1983).

[7] Johnson, W.L., "Pedagogical Agents", Proceedings of, the Sixth International Conference on Computers in Education (ICCE '98), Beijing, China, October 14 – 17, pp.13 – 22, (1998).

[8] Ju, Y. (1998), 'Development and formative evaluation of a computer-based peer tutoring system', Proceedings of, the Sixth International Conference on Computers in Education (ICCE '98), Beijing, China, October 14 – 17, pp. 559 – 566, (1998).

[9] Kinoshita, T. & Sugawara, K., "Agent oriented computing", Tokyo: Soft Research Center (1995).

[10] Laurel, B., "Interface agents: metaphors with character", in Laurel, B. (ed.) The Art of Human-Computer Interface Design, Reading, Massachusetts: Addison-Wesley (1990).

[11] Loeffler, C.E., "Artificial Life of Agents", Proceedings of the IFIP World Conference on IT Tools, Canberra, Australia, September 2 – 6, pp. 48 – 55, (1996).

[12] Maes, P., "Software Agents Tutorial", http://www.media.mit.edu/people/pattie/CHI97 (1997), (accessed on March 18, 2000).

[13] Masterton, S., "Computer support for learners using intelligent educational agents: the way forward", Proceedings of, the Sixth International Conference on Computers in Education (ICCE '98), Beijing, China, October 14 – 17, pp. 211 - 219, (1998).

[14] Minsky, M., "The Society of Mind", London: Picador/Heinemann, (1985).

[15] Pufall, P., "Function in Piaget's system: some notes for constructors of microworlds", in Forman, G. & Pufall, P. (eds) (1988), "Constructivism in the Computer Age", Hillsdale, New Jersey: Lawrence Erlbaum Associates, (1988).

[16] Sheremetov, L. & Nunez, G., "Multi-Agent Framework for Virtual Learning Spaces", Journal of Interactive Learning Research, Vol. 10, No. 3/4, pp. 235 – 243, (1999).

[17] Shirk, H.N., "Technical writers as computer scientists: the challenges of online documentation", in Barrett, E. (ed.), "Text, Context and Hypertext: writing with and for the computer", Cambridge, Massachusetts: MIT Press (1988).

[18] Solomos, K & Avouris, N. (1999), "Learning from Multiple Collaborating Intelligent Tutors: An Agent-Based Approach", Journal of Interactive Learning Research, Vol. 10, No. 3/4, pp. 235 – 243, (1999).

[19] Whatley, J., Staniford, G, Beer, M. &Scown, P. (1999), "Intelligent Agents to Support Students Working in Groups Online", Journal of Interactive Learning Research, Vol. 10, No. 3/4, pp. 235 – 243, (1999).

Generating interactive explanations by using both images and texts for Micro World

Kiyoshi Yoshikawa, Isamu Takahashi, Tatsuhiro Konishi and Yukihiro Itoh
Department of Computer Sciences, Shizuoka University
Johoku 3 - 5 - 1, Hamamatsu, Shizuoka, 432 - 8011, Japan
Tel: +81-53-478-1474 Fax: +81-53-478-1499
E-mail: {cs6098, i-takaha, konishi, itoh}@cs.inf.shizuoka.ac.jp

In this paper, we propose a method of constructing an intelligent Micro-World (MW) for high-school chemistry that accepts learners' questions and advises them whenever the learners are working in the MW. We also discuss the method generating explanations using both texts and images. At first, we argue on the interaction between a learner and such a system, and classify learner's typical demands and possible educational supports by the system. Next we show the ability necessary to deal with the demands, such as recognizing learners' plan, generating a plan to achieve a goal of an experiment, reproducing the state at any step of the change in MW, controlling the initiative of the interaction, and so on. Then we propose methods in order to realize the abilities. Moreover, we illustrate how to implement the abilities and introduce our prototype system.

Keywords: **Micro World, Interactive explanation, Mixed initiative**

1 Introduction

Micro-World (MW) has a problem that it is hard to support learners who are in impasse. We are developing an intelligent MW that supports the learners[1,2,4,5]. The domain subject of the MW is high-school chemistry. The MW has the following functions:

- Simulating changes in the world model of high-school chemistry.
- Recognizing the learner's plan by a sequence of his manipulations.
- Judging weather the learner is in impasse, by comparing the learner's plan with the standard plan that the MW generates. If the learner is in impasse, it assumes that the cause of the impasse might be lack of the knowledge necessary to perform the next manipulation which the learner should do.
- Giving the learner some advices by using texts. For example, the MW shows the knowledge which the learner doesn't understand, the manipulation that the learner should do next, and so on.
- Accepting learner's questions at any time when the learner is working in the MW• and answering by using texts.

Our MW uses only texts in giving advices. In general, it is effective to explain something by using both texts and images. CINESPEAK [3] is one of the systems which can show explanations using both images and texts. It can generate a 3D animation and texts of explanation. It also can select appropriate camera shot corresponding to the contents of explanation. However, It can not generate explanations interactively.

We think it is necessary to avoid showing the texts and images prepared beforehand like video movies with some captions. The reason is that the explanation should be shown interactively. In other words, an educational system must not explain anything one-sidedly, because the condition of a learner is changing moment by moment while the system explains to the learner.

When an idea flashes upon a learner's mind during the explanation, the system must allow him to say his idea and respond to his remark. For example, when the system explains how to solve some problems in MW to the learner who is in impasse, if the learner requires doing continuation of the problem solving process by himself, the system should prepare MW and let him continue solving the problem on MW. Similarly, if the learner requires changing some conditions of MW and explaining the method of solving the problem, the system should stop explaining, re-plan a new method to solve the problem with new conditions, and explain it.

In this paper, we extend the user interface of our MW in order to make it more effective. The first extension is that the MW uses not only texts but also images when it shows the learners advices or explanations. The second one is that the MW generates explanations interactively. Our extended system can explain manipulations that a learner performed in a MW and the manipulations necessary to achieve a given goal by using both texts and animations simultaneously. Moreover it can explain interactively according to the learner' s demand.

In the next section, we discuss the ability necessary for the system that generate explanations interactively. In section 3 we show our basic approach to realize the abilities. In section 4 we illustrate how to implement the abilities, and we introduce our prototype system and show examples of its behavior.

2 Interactive method to explain

In order to generate explanations interactively, the system should have the following two functions.
- When a learner does not express his intention, the system must be able to lead his learning.
- The system must be able to deal with a learner' s demand whenever the system aids learning (even when it is explaining something to him).

The former is out of range of this paper, because it is the topic concerning to the teaching strategy in the field of Intelligent Tutoring System (ITS). Therefore, We concentrate the latter.

Learners' demands and the method to deal with them depend on what kind of educational supports can be provided by the system. Therefore, we must clarify:
1. the educational supports and learners' demands.
2. what kinds of ability are necessary to deal with the demands.

2.1 Possible educational supports and learners' demands

We can classify states of the system into the following two types:
- The system gives a goal and the learner manipulates the MW on his own initiative.
- The system takes the initiative then it shows advice or explanations to the learner.

We discuss learner' s demands and methods to deal with them on each state.

2.1.1 Supports and learners' demands when learner has initiative

We think the major demand on this state is to require an advice to resolve a learner's impasse. Therefore, we deal with only such type of demands as the first step of our research. In order to discuss how to deal with the demands, we classify causes of learners' impasse into the following two types.
(A):• A learner cannot understand the current state of MW.
(B):• A learner cannot decide what to do in the next step.

The system can satisfy the demand of the learner who is in impasse because of (A) by showing the following explanations:
- Explanation of a sequence of manipulations that the learner performed in the MW and the effect of each manipulation.
- Explanation of the state after each manipulation has performed.

The demand of the learner who is in impasse because of (B) can be satisfied by various ways. For example, the system identifies misunderstood or lacked knowledge and shows him the knowledge, the system explains on the similar case and lets him remind his experience, and so on. In this paper, we adopt the simplest way that the system shows the actions to be performed in the following steps. If we take the other way, we need to extend some functions to decide contents of explanations. However, the mechanism to control interactive generation of explanations is commonly reused.

As a result, the type of demands of the first state is only a demand to require some advice, and the type of explanations that the system generates is only an explanation of manipulations and the state after each one. In order to explain a manipulation and the state after it has been performed, the system generates animation showing how to perform the manipulation in the MW and texts explaining the effect of the manipulation.

2.1.2 Supports and learners' demands when the system has initiative

First, we discuss typical demands of learners who are in impasse because of (A) mentioned in the previous section. When the system explains to the learner a sequence of manipulations that the learner performed and the state after each manipulation by using animations and texts, the learner may demand that the system shows him a previous state again or a following state intermittently. In case that the learner finds his own mistakes while the system is explaining something to him, he may demand that the system stops explaining, prepares the initial environment, and lets him re-try solving his problem on the MW again. If the learner fails to resolve his impasse in spite of some explanations generated by the system, he may demand that the system shows him the whole correct process to achieve his goal on the MW.

Then, we imagine typical demands of learners who are in impasse because of (B). In this case, the system explains him the action to be performed in the following steps. The learner may demand that:
- the system shows him the previous/following states.
- the system stops explaining in order to let him do continuation of manipulations.
- he rewrites some conditions of his problem and the system explains how to solve the problem with new conditions.

We don't argue on all of above-mentioned demands, but only ones with which our system can deal, considering possible actions by our system. Such actions are as follows:
(1) Explaining the sequence of actions which learners have performed.
(2) Explaining the sequence of correct actions by which the given goal can be achieved.
(3) Setting an environment for experiment to let learners try achieving the goal free.

Table 1. Examples of typical demands by learners

Actions of the system		type of the demand and the scene where the learner input the demand
Action before the demand	Action after the demand	
(1)	(1)	The learner wants to see another action than one shown in the current explanation on the sequence of his previous actions.
(1)	(2)	The learner can understand what he has done, and wants to see what he should do next.
(1)	(3)	The learner finds the mistakes he has made, and wants to re-try the experiment.
(2)	(1)	The learner understands correct actions, and wants to compare it with what he did.
(2)	(2)	The learner wants to see another action than one shown in the current explanation on the sequence of correct actions.
(2)	(3)	The learner understands the correct actions, and wants to re-try the experiment.
(3)	(1)	The learner loses the current state in the process of achieving the goal of the experiment. He wants to confirm the actions which he has performed.
(3)	(2)	The learner loses the way to achieve the goal of the experiment. He wants to see the correct actions.
(3)	(3)	He finds he has failed to achieve the goal and wants to re-try the experiment.

Then we can classify the demands according to which actions are performed before/after accepting the demand. Combinations of the actions are 3*3= 9 types such as "when system doing (1), a demand is input, then it does (1)", "when it doing (1), a demand is input, then it begins to do (2)", and so on. Examples of the typical demand of each type are shown in Table 1.

2.2 Abilities necessary to deal with learners' demands

In this section, we discuss abilities necessary to deal with the learners' demands mentioned in 2.1. Basically, MW should have an ability to simulate changes in the MW according to learners' actions.
In addition, in order to deal with the demands mentioned in 2.1.1, the system should have the following abilities.

(a) Ability to recognize learners' plan from a sequence of his actions.

In order to explain what learners have done by not only listing up the actions, but also showing the meanings of the sequence of the actions, the system needs the ability.

(b) Ability to generate a plan to achieve a goal of an experiment.

In order to explain correct actions which learners should perform, the system has to be able to generate plan.

(c) Ability to simulate changes in the MW according to the plan generated or recognized by itself, and ability to generate verbal explanations showing what actions has been done or what actions should be going to be done.

The system had better be able to generate explanations using both texts and images. In order to generate visual explanations, the system should be able to operate MW in a similar way as learners do. In order to generate verbal explanations, the system should be able to generate texts from the result of planning or plan recognition.

In order to deal with the demands mentioned in 2.1.2, the abilities mentioned above are also necessary. In addition, the following abilities are needed.

(d) Ability to store the history of actions by learners or the system.

The ability is needed to do action (1) or (2) as a reaction of a demand in Table 1.

(e) Ability to reproduce the state at any step of the change in MW and allow learners to manipulate the MW.

The ability is needed to do action (3) as a reaction of a demand in Table 1.

In addition, the following ability is necessary to realize mixed initiative. It is generally important to make interactive educational environment effective.

(f) Ability to control the two phases: a phase where a learner takes initiative by actions to achieve the goal, and a phase where the system takes initiative by generating explanations.

3 Methods necessary to deal with learner's demand

The basic framework of the system as a MW can be seen in [5]. An extension in this paper is that the system becomes to have two individual environments: one is the environment for experiment used by learners, and the other is the environment for explanation. Our system operates the latter environment in its explanation. We add the latter environment in order to avoid that both a learner and the system try to operate a common one at the same time. The environment for experiment has an interface and functions to accept learner's actions, and reacts as soon as it accepts an action from a learner. On the other hand, the environment for explanation cannot accept manipulations from learners (though switches similar to the environment of experiment are displayed in its window, they are dummy).

We discuss how to equip such a framework of the system with the abilities mentioned in 2.2.

(a) Ability to recognize learners' plan from a sequence of his actions.

On this ability, please see our previous paper• 1• ;

(b) Ability to generate a plan to achieve a goal of an experiment.

On this ability, please see our previous paper• 1• ;too.

(c) Ability to simulate changes in the MW according to the plan generated or recognized by itself, and ability to generate verbal explanations showing what actions has been done or what actions should be going to be done.

Simulation in MW is performed by using symbolic knowledge representation. States at each step of

MW are also represented in a symbolic way. Manipulations by learners are also translated to symbolic representations. The control method of the simulator is event-driven: as soon as a manipulation is input to the simulator, the inference engine generates symbolic representation showing the next state of MW. The system draws the state of MW on the basis of the symbolic representation. Therefore, the system can simulate changes in the MW according to the generated or recognized plan, because the system can generate the input of the simulator represented symbolically from the plan.

In addition, because states of MW, manipulations to MW, and changes in MW are commonly represented in a symbolic way, the system can generate explanations in natural language on every fact in MW.

(d) Ability to store the history of actions by learners or the system.

It is easy to store such history because all of states of MW, manipulations to MW, and changes in MW are represented in a symbolic way. The system records only the initial state and a sequence of having performed actions as the history. The system can reproduce all states and changes by simulating the change in MW again on the basis of the history.

(e) Ability to reproduce the state at any step of the change in MW and allow learners to manipulate the MW.

The system can reproduce any states in an explanation on learner's previous actions, by performing the manipulations stored as the history sequentially. On the other hand, it can also reproduce any states in the process when correct actions are performed, by performing the manipulations in the plan generated by itself. Thus, the system can set any states of an environment which learners can manipulate, by copying such reproduced states in the environment for explanation to the one for experiment.

(f) Ability to control the two phases: a phase where a learner takes initiative by actions to achieve the goal, and a phase where the system takes initiative by generating explanations.

We adopt the following strategies for controlling the phases:

- Basically, a learner takes initiative, and he acts freely in MW.
- Turn over the phase to the other phase where the system takes initiative, as soon as the learner inputs a question or demands that the system explains something.
- If the system finds that the learner is in impasse, ask him whether he hopes to turn over the phase where the system takes initiative. And if he does, turn over it.
- Accept interruption by learners whenever the system generates explanations.
- Decide the next action of the system according to the interruption. For example, if the learner demands that the system sets the phase where the learner takes initiative, set a suitable state of the environment and let him experiment freely. If he inputs a demand for the system to explain other topic than the current topic, continue explanation on the requested topic.

4 Implementation

We designed a prototype system. Figure 1 shows outline of our system.

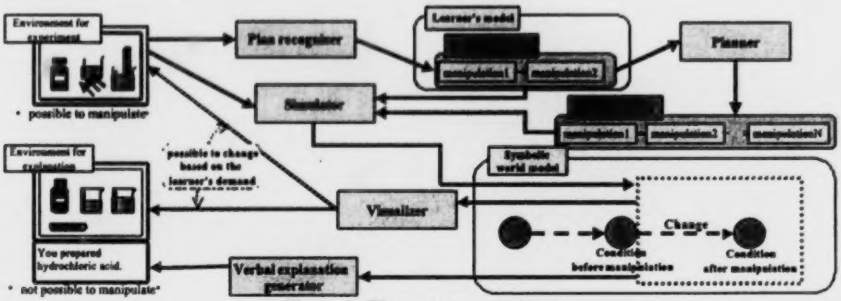

Environment for experiment
possible to manipulate
Plan recognizer
Learner's model
manipulation1
manipulation2
Planner
Simulator
manipulation1
manipulation2
manipulationN
Environment for explanation
possible to change based on the learner's demand
Symbolic world model
Visualizer
Change
Condition before manipulation
Condition after manipulation
You prepared hydrochloric acid.
not possible to manipulate
Verbal explanation generator

Figure 1: The prototype system

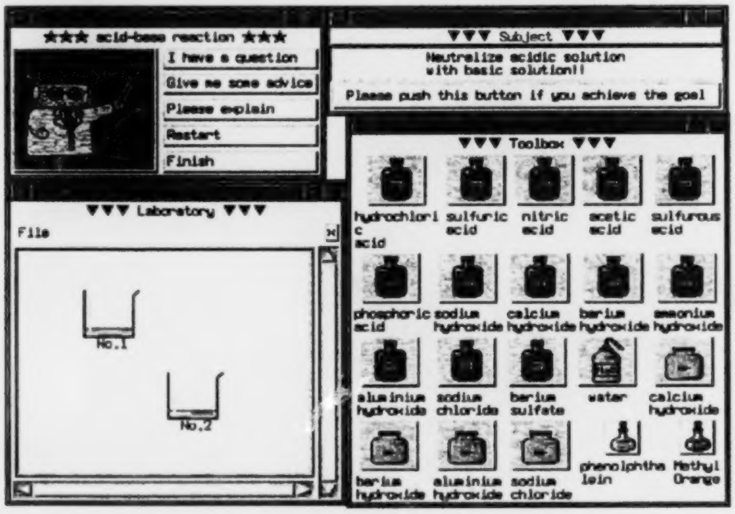
BEST COPY AVAILABLE

The system has *environment for experiment* and *environment for explanation*. The system sets a goal and a learner tries to achieve the goal by manipulating objects in *environment for experiment*. When the learner does an action in *environment for experiment*, the simulator reproduces a change in *symbolic world model*. Then the *visualizer* draws the state after the change in *environment for experiment*. At the same time, *the plan recognizer* monitors the learner' s manipulations and recognizes his plan. When the learner becomes to be in impasse or requires some advice, *planner* generates a correct plan. Then the system visualizes environment for explanation, and starts explaining by using either recognized plan or the correct plan. In order to generate explanation, *simulator* reproduces states of the world model and *visualizer* visualizes the states in *environment for explanation*. Simultaneously *Verbal explanation generator* generates verbal explanation on the manipulation, the change, and the state.

The domain world model of this system is written by symbolic representation. In general, it is difficult to handle continuity of time and space by such representation. Therefore, our system handles time as a sequence of discrete segments of time. And it doesn' t handle strict position of objects in the world, but only relative relations which can be represented by symbols, such as "chemical materials are in the same beaker". A change is also represented by symbols which shows the initial state, the actions causing the change, the changing state, and the state after the change. Most of the subjects in high-school chemistry can be handled in the above mentioned way.

This system is implemented by using Tcl/Tk and LISP (Kyoto Common LISP). This system can deal with the 5 subjects: method of preparing a solution of a certain molarity, acidic material, basic material, neutralization, and using indicator.

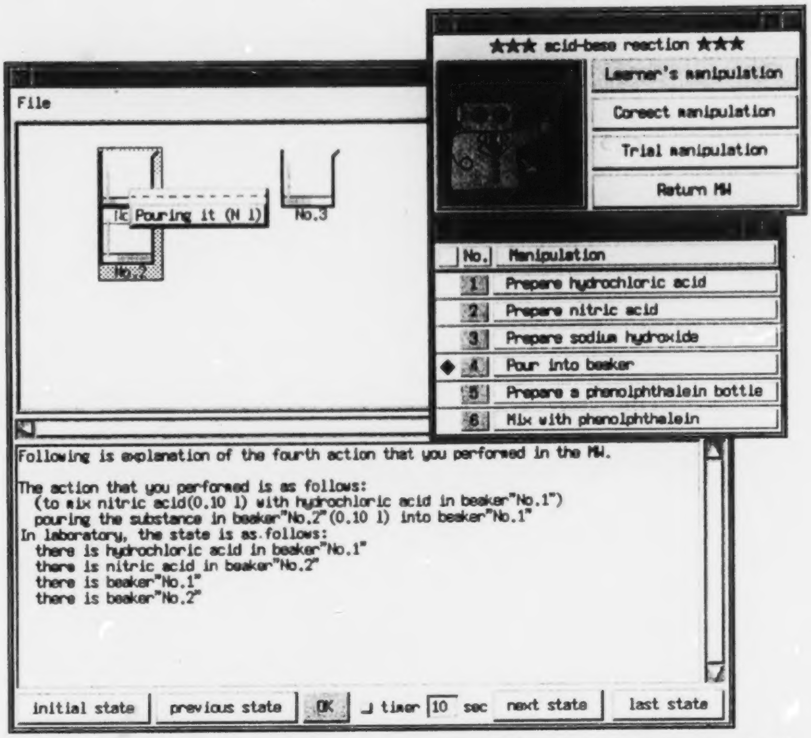
We show an example of the behavior of our system when a learner does an experiment of neutralization. Figure 2 shows a user interface for *environment for experiment*. In the environment, the learner prepares hydrochloric acid, prepares nitric acid, and sodium hydroxide, pours nitric acid into hydrochloric acid, prepares phenolphthalein, and mixes it into the mixed acid. Then he finds that he has not achieved his goal. In the case that he can' t find the reason and inputs a demand for the system to explain his own actions, the system prepares an *environment for explanation* to start explaining the actions the learner has done. Figure 3 shows a user interface of *environment for explanation*. The interface has three windows: a window for displaying visual explanations and verbal explanations, a window showing a history of actions that have been taken place, and a window for inputting demands. In Figure 3, both visual and verbal explanations for the fourth action (marked in the list shown in the window for history). If the learner finds that he has made a mistake, and if he cannot find the correct way, he wants to demand that the system explains how to neutralize. He clicks on the button "*correct manipulation*" in the window for inputting demands. Then the system starts explaining the correct way (Figure 4). If he inputs a demand for the system to let him re -try the experiment, the system prepares *environment for experiment* and reproduces the state from which he wants to start experiment (Figure 5).


★★★ acid-base reaction ★★★
I have a question
Give me some advice
Please explain
Restart
Finish
▼▼▼ Subject ▼▼▼
Please push this button if you achieve the goal
▼▼▼ Laboratory ▼▼▼
File
No.1
No.2
▼▼▼ Toolbox ▼▼▼
hydrochloric acid
sulfuric acid
nitric acid
acetic acid
sulfurous acid
phosphoric acid
sodium hydroxide
calcium hydroxide
barium hydroxide
ammonium hydroxide
aluminium hydroxide
sodium chloride
barium sulfate
water
calcium hydroxide
barium hydroxide
aluminium hydroxide
sodium chloride
phenolphthalein
Methyl Orange

Neutralize acidic solution
with basic solution!!

Figure 2: An example of environment for experiment

BEST COPY AVAILABLE


★★★ acid-base reaction ★★★
Learner's manipulation
Correct manipulation
Trial manipulation
Return MW
File
Pouring it (N 1)
No.3
No.2
No.
Manipulation
1 Prepare hydrochloric acid
2 Prepare nitric acid
3 Prepare sodium hydroxide
◆ 4 Pour into beaker
5 Prepare a phenolphthalein bottle
6 Mix with phenolphthalein
initial state
previous state
OK
☐ timer 10 sec
next state
last state

```
Following is explanation of the fourth action that you performed in the MW.

The action that you performed is as follows:
  (to mix nitric acid(0.10 l) with hydrochloric acid in beaker"No.1")
  pouring the substance in beaker"No.2"(0.10 l) into beaker"No.1"
In laboratory, the state is as follows:
  there is hydrochloric acid in beaker"No.1"
  there is nitric acid in beaker"No.2"
  there is beaker"No.1"
  there is beaker"No.2"
```

Figure 3: An example of environment for explanation (Explanations of a learner' s actions)

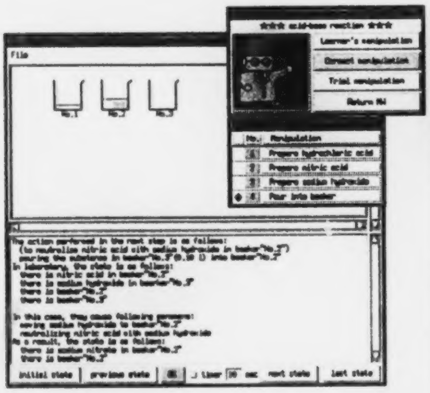
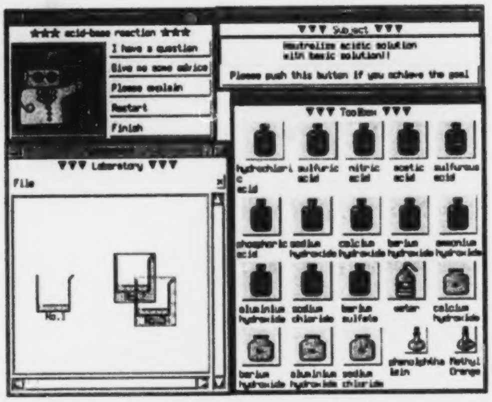

File
Trial manipulation
initial state
previous state
timer
next state
last state

Figure 4: Explanations of corrects actions


★★★ acid-base reaction ★★★
I have a question
Give me some advice
Please explain
Restart
Finish
▼▼▼ Subject ▼▼▼
Please push this button if you achieve the goal
▼▼▼ Laboratory ▼▼▼
File
No.1
▼▼▼ ToolBox ▼▼▼
hydrochloric acid
sulfuric acid
nitric acid
acetic acid
sulfurous acid
phosphoric acid
sodium hydroxide
calcium hydroxide
barium hydroxide
ammonium hydroxide
aluminium hydroxide
sodium chloride
barium sulfate
water
calcium hydroxide
barium hydroxide
aluminium hydroxide
sodium chloride
phenolphthalein
Methyl Orange

Neutralize acidic solution
with basic solution!!

Figure 5: A reproduced environment for experiment

5 Conclusions

In this paper, we discuss a method constructing an intelligent and interactive MW generating explanations both images and texts. Our prototype system has relatively small domain knowledge base, so we have to make it larger in the future in order to increase subjects that our system can support.

When we will try to extend our system to handle other domains, the simulator underlying the system needs to deal with continuity of time and space. For example, if we deal with the field of electric circuit, the

BEST COPY AVAILABLE

simulator needs to handle topology. If we deal with the field of dynamics of physics, the simulator needs to handle coordinate system.

Our another future work is to evaluate the effectiveness of our system experimentally.

Acknowledgements

This research was supported in part by Grant-in-Aid for Encouragement of Young Scientists (12780125) from Japan Society for the Promotion of Science.

References

[1] Itoh,Y., Takahashi,I. and Konishi,T., "On Recognition of Student's Actions in ILE", Proceedings of the International Conference on Computers in Education(ICCE'95), Singapore, December5-8, pp.630-639,(1995)

[2] Itoh,Y., Takahashi,I. and Konishi,T., "On evaluating students' knowledge by observing their actions in ILE", Proceeding of Artificial Intelligence in Education(AI-ED'97), Kobe,Japan, August18-22, pp.595-597, (1997)

[3] Stuart G. Towns, Charles B. Callaway and James C. Lester, " Generating Coordinated Natural Language and 3D Animations for Complex Explanations", Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, America, July 26-30, pp.112-119, (1998)

[4] Takahashi,I., Itoh,Y. and Konishi,T., "On a method to decide the order of knowledge shown as advice in ILE", Proceedings of the Sixth International Conference on Computers in Education (ICCE'98), Beijing, China, October 14-17, vol1, pp.373-380, (1998)

[5] Takahashi,I., Konishi,T. and Itoh,Y., "On a Framework of an Educational System which Acts both as ITS and Micro-World", Proceedings of the Seventh International Conference on Computers in Education (ICCE'99), Chiba, Japan, November 4-7, pp.954-957, (1999)

Intelligent Interactive Learning Environment: Design Issues

Siu-Cheung KONG* and Lam-For KWOK**
**Department of Information & Applied Technology, Hong Kong Institute of Education, 10, Lo Ping Road, Tai Po, N.T.,Hong Kong. Email: sckong@ied.edu.hk*
***Department of Computer Science,City University of Hong Kong,Tat Chee Avenue, Kowloon, Hong Kong. Email: cslfkwok@cityu.edu.hk*

Interactive Learning Environment (ILE) provides interaction opportunities between learners and the virtual devices for productive learning. Intelligent ILE (IILE) provides quality feedback or authentic guidance to learners who need help in the ILE. This research aims to explore design implications of IILE by studying model of learner in the mathematics fraction domain. 169 primary four learners were invited to answer 10 open-ended questions on fraction addition and subtraction. A learner model on category of error and error pattern was formulated from the 423 erroneous responses. Results of the study indicated that researchers should study error patterns by understanding work of learners, distinguish careless mistakes from error patterns, and consider scaffolding support.

Keywords: **Intelligent Interactive Learning Environment, Learner Model**

1 INTRODUCTION

There are two categories of Learning Environment (LE): content-free and subject-specific [1]. A content-free LE allows participants and facilitators to formulate their own topics for discussion. Knowledge formulated from such interactions belongs to the learning community [2]. A subject-specific LE involves subject knowledge. Some subject-specific environments stress knowledge transfer like Intelligent Tutoring System (ITS) [3]. Other subject-specific environments such as Interactive Learning Environments (ILE), assisting learners to learn through exploration, put efforts on designing manipulative virtual learning devices [4]. No matter an LE is designed for knowledge transfer or knowledge formulation, subject matter of the learning domain should be carefully studied and incorporated in it [5].

1.1 Design Considerations of an ILE

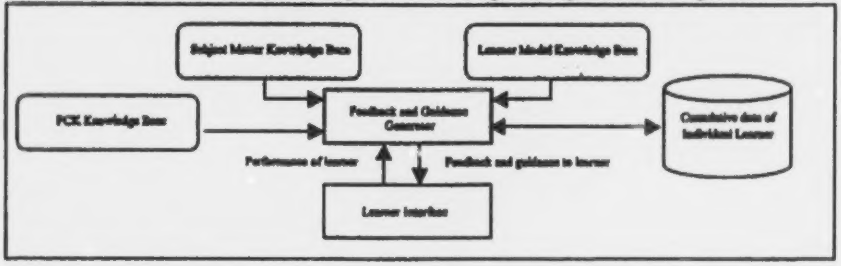
The study of subject matters plays a crucial role in designing ILE involving knowledge exploration because learners are not obtaining knowledge directly from the ILE. Learners have to learn by analogy, that is, learners have to transfer knowledge from manipulating the manipulative virtual devices of the ILE to grasp the abstract concepts of the subject domain [4]. Expert teachers are skilful in predicting how learners will think and err [6]. This diagnostic ability is tied to an expert's special understanding of the subject and is undoubtedly derived from multiple opportunities to teach the same content [7]. This knowledge includes knowing which aspects of a topic are particularly difficult, what the common misconceptions are, and what representations are important for authentic learning. Shulman [8] termed this kind of knowledge as Pedagogical Content Knowledge (PCK). It is crucial to utilize teachers' expert knowledge, especially knowledge on representation for authentic learning, to design manipulative virtual devices of an ILE.

1.2 Design Considerations of an Intelligent ILE

An ILE may provide interaction opportunities between learners and the virtual devices for productive learning. Some learners may learn the subject matter well without the assistance of the virtual learning devices. Some learners may learn well with chances to interact with the interactive learning devices of the environment. However, some learners may need guidance to learn well in the ILE [9]. An Intelligent ILE (IILE) is an ILE that provide feedback or guidance to those learners who need such help in learning the subject domain. Those learners who do not need help will not notice the existence of the auxiliary service. Learner model of learning in a subject domain may provide information about the behaviour of learners in learning the domain. Studying the learning model of learners may assist IILE designers to formulate design principles and obtain technical details such as formulating mal rules for understanding learning states of learners. A learner model thus may help to tailor-make an IILE for assisting various types of learners in learning the discipline. It is therefore important to study the learning model of learners in a specific subject domain for designing a useful and practical IILE to assist learners of various kinds in the learning process.

Three knowledge bases are therefore important for designing an IILE for learning subject-specific knowledge. They are the subject matter, the learner model of learning in the domain and the PCK of teachers in teaching the discipline. Subject matter knowledge base contains subject matter knowledge. It can provide subject matter advice and knowledge state of learners in the learning process. Learner model contains behaviour representations of learners. Learner model knowledge base may provide information about the learning state of learner. PCK knowledge base contains diverse guidance knowledge for different learning states of learners. It may provide learning advises based on PCK of experienced teachers of the subject domain who know how learners think and err in the discipline. Software agents will monitor the performance of learner in the learner interface. Software agents will determine proactive or reactive responses after a negotiation and communication process in the feedback and guidance generator. The negotiation will be a judgement of the knowledge state of the learner in the domain using both the learner model knowledge base and subject matter knowledge base of the IILE. Final decision will be an outcome after a consultation with the PCK knowledge base of the IILE and the cumulative data of an individual learner. The cumulative data records the historical learning states of each individual learner captured by the IILE. Figure 1 shows a conceptual design of an IILE for generating feedback and guidance.

Figure 1: A conceptual design of an IILE for generating feedback and guidance


Subject Matter Knowledge Base
Learner Model Knowledge Base
PCK Knowledge Base
Feedback and Guidance Generator
Cumulative data of Individual Learner
Performance of learner
Feedback and guidance to learner
Learner Interface

1.3 Chosen Subject Domain

A review of literatures indicated that many learners have great difficulties in learning the concepts and procedural knowledge of mathematics fraction [10, 11, 12]. Streefland [11] further pointed out that the main cause of such difficulties is the inadequate and inappropriate teaching in the traditional approaches. As the teaching and learning of mathematics fraction is an internationally renowned difficult topic, it is considered as an appropriate exemplar to be investigated for automation.

2 AIM AND OBJECTIVES

The aim of this research is to study the knowledge of learners in a subject-specific domain and to investigate its implication for designing a subject-specific IILE. There are two specific objectives: (1) to understand the problems of learners in learning the topic; (2) to discuss design issues of an IILE. Such findings may inform the development of IILE for providing quality feedback and guidance to learners.

3 RESEARCH METHODOLOGY

A questionnaire for studying model of primary learners on learning fraction addition and subtraction was designed.

169 primary four learners from four different schools were invited to complete the questionnaire through their mathematics teachers. All learners had completed their learning of fraction addition and subtraction before the test. Learners were requested to do the questionnaire on individual basis in a mathematics lesson for about 35 minutes. No discussions were allowed. The answer sheets were not used for any form of assessment but returned to the researcher after the administration. All 169 answer sheets returned were used for data analysis.

4 RESULTS AND DISSCSSIONS

This section will report on the quantitative and qualitative analysis results of all errors responded by participants of the survey and will discuss their implications on designing an IILE. The learner model formulated contains two areas: (1) knowledge of learners on category of error; and (2) knowledge of learners on error patterns of the domain.

4.1 Knowledge of Learner on Category of Error

Nine categories of error were identified and summarized from the 423 incorrect responses. Though incorrect response of each question may contain more than one error, this study selected the primary source of error for classification. Results were summarized in table 1. Categories were organized in descending order of percentage that account for the errors. The summarized result may serve as an important reference in designing a learner model of LE for fraction learning. Among the nine categories, categories 1, 2 and 9 directly related to the subject matter and accounted for nearly forty percent of the erroneous work. Categories 3 and 8 were common types of error in any mathematics exercise. It is interesting to investigate whether learners in this age group would commit these types of error like doing subtraction for addition at a certain level of unconsciousness. The study reflected that these factors might account for another twenty percents of errors.

Table 1: Category of error summarized from the learner model of the study

	Category of Error	Percentage Accounted
1	Improper handling of mixed number in fraction operation	20.4%
2.	Insufficient procedural knowledge for evaluating fraction	14.7%
3.	Calculation or careless mistake	13.5%
4	Unable to set up correct expression for solving word problem	11.6%
5.	Incorrect strategy for evaluating expression	11.4%
6.	Unable to identify error pattern for erroneous work	10.9%
7.	Not responding to question or the piece of work unfinished	8.5%
8.	Conducting subtraction for addition and similarly addition for subtraction	5.5%
9.	Incorrect simplification of answer to the simplest fraction form	3.6%

Though categories 4 and 5 can be purposely avoided, they do play a role in mathematics learning. Setting up expression for solving problems in a scenario may help to test whether a learner has grasped the taught concept. Strategies of evaluating numerical expressions may help to detect whether a learner has knowledge on magnitude of operands and order of evaluation on operators in an expression. The deficiency of this knowledge accounted for twenty percents of errors detected in this study. Categories 6 and 7 accounted for the last twenty percent of learners' work that might not be understandable or remain unfinished. Those 10 percent of learners' work could not be identified for any error pattern reflected that even human teachers might be unable to understand open-ended pieces of work like evaluating mathematics expressions.

4.2 Knowledge of Learner on Error Patterns

This section will report on knowledge of learners with problems in working with fractions on addition and subtraction. After careful analysing error patterns of learners in evaluating and solving simple fraction addition and subtraction problems, two categories were summarized: (1) concrete error pattern; and (2) vague idea on working with fractions. The first category includes some concrete error patterns that can be abstracted into mal rules. The second category contains error patterns that cannot be easily summarized into mal rules but reflect vague ideas and incomplete working procedures of learners. One of the most famous mal rules on fraction addition can be named as "Add numerators and add denominators". Learner with poor knowledge on fraction addition will adopt knowledge of arithmetic addition by adding the numerators of fractions in the fraction expression to give the numerator of the resultant fraction and similarly adding the denominators of fractions to give the denominator of the resultant fraction. There were four learners committing this type of error in this study. This rule might explain 3% of the errors. The second category of error pattern to be analysed involves high-level abstraction. The group of learners in this category showed no concrete error patterns. However, the pattern illustrated that these learners have some vague ideas of doing fraction addition and subtraction. Examples were illustrated in table 2.

Table 2 Vague ideas for evaluating fraction addition and subtraction expressions

	Error 1	Error 2
Learner 1 (3 score)	$\frac{3}{8}+\frac{1}{6}=\frac{9}{18}+\frac{9}{18}=\frac{18}{18}=1$	
Learner 2 (6 score)	$\frac{1}{2}+\frac{1}{3}=\frac{3}{10}-\frac{2}{10}=\frac{5}{10}=\frac{1}{2}$	
Learner 3 (0 score)	$\frac{1}{2}+\frac{1}{3}=\frac{1\times3}{2\times3}+\frac{1\times3}{3\times1}=\frac{3}{6}+\frac{2}{6}=\frac{6}{5}$	$\frac{3}{8}+\frac{1}{6}=\frac{3\times6}{8\times6}+\frac{1\times6}{3\times6}=\frac{54}{18}+\frac{3}{18}=\frac{12}{18}=\frac{6}{5}$

These erroneous presentations reflected that learners did have vague ideas about the working procedures on fraction addition. They need assistance to organize the disconnected nodes into a semantic net. Result of the studies indicated that some error patterns could be represented by mal rules. However, there were even more that cannot. An alternate method of studying error patterns of learners is to understand their work.

Identify Careless Mistake

The learner model of this study reflected that twenty percent of errors were derived from calculation or careless mistakes. Careless mistakes in this study mean transcription errors or simple computational mistakes form one step to another. The feedback and guidance will be different if an error is identified as a careless one. An IILE should handle not only problems generated from subject matters but also general problems of learner like careless mistake. An authentic guidance should provide not only advices or actions that can assist learners to formulate conceptual understanding of the subject domain but also offer help to learners derived from general problems such as careless mistakes. An IILE should attempt to distinguish careless mistake from other error patterns like human teachers.

Scaffolding Support

The forty percent of errors derived from inadequate knowledge of learners reflected that only immediate feedback may not help learner much and thus authentic guidance should be considered for facilitating conceptual understanding. A productive learning support should be an arrangement of a sequence of situations for facilitating knowledge construction [12]. The role of a mathematics-learning environment will be to help learners to learn, especially those fundamental concepts in mathematics, but not to replace mathematics learning in the conventional manner. Therefore it is fundamental for such kind of learning environment to provide scaffolding support to learner when assistance is needed. Support should gradually withdraw so that learner can stand on its own after leaving the system. Therefore a fraction IILE should be designed like a blank sheet for learner to work with fraction. Feedback and guidance are only provided when it is needed. On the other hand, learner working in the IILE who does not need support will not notice the IILE in behind.

5 CONCLUSION

Studying the learning model of learners may assist IILE designers to formulate design principles and obtain details for understanding learning states of learners. The learner model of this study modelled behaviour of learners in two aspects: error category and error patterns. Nine categories of error were identified. Forty percent of errors were derived from inadequate knowledge of learners on subject matters. Twenty percent could be explained by careless mistakes. Twenty percent involved general mathematics knowledge. The final twenty percent of erroneous work were difficult to be classified or work was not completed. Learner model of the study reflected that some error patterns could be represented by mal rules. However, there were even more that cannot. An alternate method of studying error patterns of learners is to understand their work. Result of the study indicated that IILE needed to apply a strategy to identify careless mistake so that appropriate guidance to learners can be provided. The forty percent of errors derived from inadequate knowledge of learners reflected that only immediate feedback may not help much and thus authentic guidance should be considered for facilitating conceptual understanding. A productive scaffolding support should be an arrangement of situations for facilitating knowledge construction. The future work of the study is to design ways and means to understand work of students, to devise strategy to distinguish careless mistake from other error patters, and to plan scenarios for assisting learners to learn by exploration in an IILE.

References

[1] Vosniadou, S., Corte, E.D., Glaser, R., & Mandl, H. (Eds.), "International perspectives on the design of technology-supported learning environments", Hillsdale, NJ: Lawrence Erlbaum, (1996).

[2] Duffy, T., & Jonassen, D. (Eds.), "Constructivism and the technology of instruction: A conversation", Hillsdale, NJ: Lawrence Erlbaum, (1992).

[3] Chan, T.W., "An introduction to intelligent computer assisted learning systems research and development", National Science Council Monthly, Vol. 23, No. 5, pp. 456-468, (1995).

[4] Akpinar, A., & Hartley, J.R., "Designing interactive learning environments", Journal of Computer Assisted Learning, Vol. 12, pp. 33-46, (1996).

[5] Wittmann, E.Ch., "Mathematics education as a 'design science' ", Mathematics education as a research domain: A search for identity, Dordrecht: Kluwer Academic Publishers, pp. 87-103, (1998).

[6] Berliner, D.C., "In pursuit of the expert pedagogue. Educational Researcher", Vol. 15, No. 7, pp. 5-13, (1986).

[7] Livingston, C., & Borko, H., "Expert-novice differences in teaching: A cognitive analysis and implications for teacher education", Journal of Teacher Education, Vol. 40, No. 4, pp. 36-42, (1989).

[8] Shulman, L.S., "Those who understand: Knowledge growth in teaching", Educational Researcher, Vol. 15, No. 2, pp. 4-14, (1986).

[9] Cheng, C.C., "Proactive guidance in computer-assisted language learning", Technical Report, LLL-T-24-95. Language Learning Laboratory, College of Liberal Arts and Sciences, University of Illinois, (1995).

[10] Behr, M.J., Harel, G., Post, T. & Lesh, R., "Rational number, ratio and proportion", Handbook of research on mathematics teaching and learning, New York: Macmillan, pp. 296-333, (1993).

[11] Streefland L., "Fractions in realistic mathematics education: A paradigm of developmental research", Dordrecht: Kluwer Academic Publishers, (1991).

[12] Ohlsson, S., "Knowledge requirements for teaching: The case of fractions", Teaching knowledge and intelligent tutoring, Norwood, NJ: Ablex Publishing Corporation, pp. 25-59, (1991).

Intelligent Interactive Learning Environment: Design Issues

Siu-Cheung KONG* and Lam-For KWOK**
**Department of Information & Applied Technology, Hong Kong Institute of Education, 10, Lo Ping Road, Tai Po, N.T.,Hong Kong. Email: sckong@ied.edu.hk*
***Department of Computer Science,City University of Hong Kong,Tat Chee Avenue, Kowloon, Hong Kong. Email: cslfkwok@cityu.edu.hk*

Interactive Learning Environment (ILE) provides interaction opportunities between learners and the virtual devices for productive learning. Intelligent ILE (IILE) provides quality feedback or authentic guidance to learners who need help in the ILE. This research aims to explore design implications of IILE by studying model of learner in the mathematics fraction domain. 169 primary four learners were invited to answer 10 open-ended questions on fraction addition and subtraction. A learner model on category of error and error pattern was formulated from the 423 erroneous responses. Results of the study indicated that researchers should study error patterns by understanding work of learners, distinguish careless mistakes from error patterns, and consider scaffolding support.

Keywords: **Intelligent Interactive Learning Environment, Learner Model**

1 INTRODUCTION

There are two categories of Learning Environment (LE): content-free and subject-specific [1]. A content-free LE allows participants and facilitators to formulate their own topics for discussion. Knowledge formulated from such interactions belongs to the learning community [2]. A subject-specific LE involves subject knowledge. Some subject-specific environments stress knowledge transfer like Intelligent Tutoring System (ITS) [3]. Other subject-specific environments such as Interactive Learning Environments (ILE), assisting learners to learn through exploration, put efforts on designing manipulative virtual learning devices [4]. No matter an LE is designed for knowledge transfer or knowledge formulation, subject matter of the learning domain should be carefully studied and incorporated in it [5].

1.1 Design Considerations of an ILE

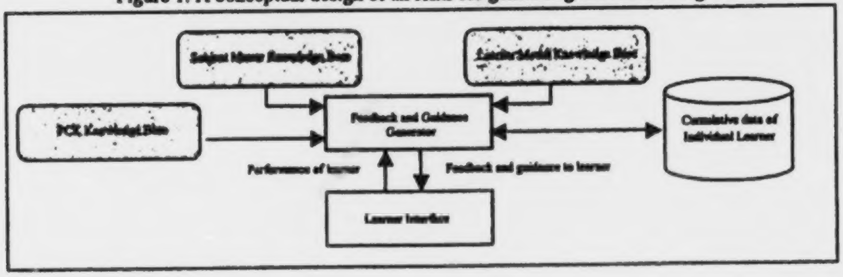
The study of subject matters plays a crucial role in designing ILE involving knowledge exploration because learners are not obtaining knowledge directly from the ILE. Learners have to learn by analogy, that is, learners have to transfer knowledge from manipulating the manipulative virtual devices of the ILE to grasp the abstract concepts of the subject domain [4]. Expert teachers are skilful in predicting how learners will think and err [6]. This diagnostic ability is tied to an expert's special understanding of the subject and is undoubtedly derived from multiple opportunities to teach the same content [7]. This knowledge includes knowing which aspects of a topic are particularly difficult, what the common misconceptions are, and what representations are important for authentic learning. Shulman [8] termed this kind of knowledge as Pedagogical Content Knowledge (PCK). It is crucial to utilize teachers' expert knowledge, especially knowledge on representation for authentic learning, to design manipulative virtual devices of an ILE.

1.2 Design Considerations of an Intelligent ILE

An ILE may provide interaction opportunities between learners and the virtual devices for productive learning. Some learners may learn the subject matter well without the assistance of the virtual learning devices. Some learners may learn well with chances to interact with the interactive learning devices of the environment. However, some learners may need guidance to learn well in the ILE [9]. An Intelligent ILE (IILE) is an ILE that provide feedback or guidance to those learners who need such help in learning the subject domain. Those learners who do not need help will not notice the existence of the auxiliary service. Learner model of learning in a subject domain may provide information about the behaviour of learners in learning the domain. Studying the learning model of learners may assist IILE designers to formulate design principles and obtain technical details such as formulating mal rules for understanding learning states of learners. A learner model thus may help to tailor-make an IILE for assisting various types of learners in learning the discipline. It is therefore important to study the learning model of learners in a specific subject domain for designing a useful and practical IILE to assist learners of various kinds in the learning process.

Three knowledge bases are therefore important for designing an IILE for learning subject-specific knowledge. They are the subject matter, the learner model of learning in the domain and the PCK of teachers in teaching the discipline. Subject matter knowledge base contains subject matter knowledge. It can provide subject matter advice and knowledge state of learners in the learning process. Learner model contains behaviour representations of learners. Learner model knowledge base may provide information about the learning state of learner. PCK knowledge base contains diverse guidance knowledge for different learning states of learners. It may provide learning advises based on PCK of experienced teachers of the subject domain who know how learners think and err in the discipline. Software agents will monitor the performance of learner in the learner interface. Software agents will determine proactive or reactive responses after a negotiation and communication process in the feedback and guidance generator. The negotiation will be a judgement of the knowledge state of the learner in the domain using both the learner model knowledge base and subject matter knowledge base of the IILE. Final decision will be an outcome after a consultation with the PCK knowledge base of the IILE and the cumulative data of an individual learner. The cumulative data records the historical learning states of each individual learner captured by the IILE. Figure 1 shows a conceptual design of an IILE for generating feedback and guidance.

Figure 1: A conceptual design of an IILE for generating feedback and guidance


Subject Matter Knowledge Base
Learner Model Knowledge Base
PCK Knowledge Base
Feedback and Guidance Generator
Cumulative data of Individual Learner
Performance of learner
Feedback and guidance to learner
Learner Interface

1.3 Chosen Subject Domain

A review of literatures indicated that many learners have great difficulties in learning the concepts and procedural knowledge of mathematics fraction [10, 11, 12]. Streefland [11] further pointed out that the main cause of such difficulties is the inadequate and inappropriate teaching in the traditional approaches. As the teaching and learning of mathematics fraction is an internationally renowned difficult topic, it is considered as an appropriate exemplar to be investigated for automation.

2 AIM AND OBJECTIVES

The aim of this research is to study the knowledge of learners in a subject-specific domain and to investigate its implication for designing a subject-specific IILE. There are two specific objectives: (1) to understand the problems of learners in learning the topic; (2) to discuss design issues of an IILE. Such findings may inform the development of IILE for providing quality feedback and guidance to learners.

3 RESEARCH METHODOLOGY

A questionnaire for studying model of primary learners on learning fraction addition and subtraction was designed.

BEST COPY AVAILABLE

169 primary four learners from four different schools were invited to complete the questionnaire through their mathematics teachers. All learners had completed their learning of fraction addition and subtraction before the test. Learners were requested to do the questionnaire on individual basis in a mathematics lesson for about 35 minutes. No discussions were allowed. The answer sheets were not used for any form of assessment but returned to the researcher after the administration. All 169 answer sheets returned were used for data analysis.

4 RESULTS AND DISSCSSIONS

This section will report on the quantitative and qualitative analysis results of all errors responded by participants of the survey and will discuss their implications on designing an IILE. The learner model formulated contains two areas: (1) knowledge of learners on category of error; and (2) knowledge of learners on error patterns of the domain.

4.1 Knowledge of Learner on Category of Error

Nine categories of error were identified and summarized from the 423 incorrect responses. Though incorrect response of each question may contain more than one error, this study selected the primary source of error for classification. Results were summarized in table 1. Categories were organized in descending order of percentage that account for the errors. The summarized result may serve as an important reference in designing a learner model of LE for fraction learning. Among the nine categories, categories 1, 2 and 9 directly related to the subject matter and accounted for nearly forty percent of the erroneous work. Categories 3 and 8 were common types of error in any mathematics exercise. It is interesting to investigate whether learners in this age group would commit these types of error like doing subtraction for addition at a certain level of unconsciousness. The study reflected that these factors might account for another twenty percents of errors.

Table 1: Category of error summarized from the learner model of the study

	Category of Error	Percentage Accounted
1	Improper handling of mixed number in fraction operation	20.4%
2.	Insufficient procedural knowledge for evaluating fraction	14.7%
3.	Calculation or careless mistake	13.5%
4	Unable to set up correct expression for solving word problem	11.6%
5.	Incorrect strategy for evaluating expression	11.4%
6.	Unable to identify error pattern for erroneous work	10.9%
7.	Not responding to question or the piece of work unfinished	8.5%
8.	Conducting subtraction for addition and similarly addition for subtraction	5.5%
9.	Incorrect simplification of answer to the simplest fraction form	3.6%

Though categories 4 and 5 can be purposely avoided, they do play a role in mathematics learning. Setting up expression for solving problems in a scenario may help to test whether a learner has grasped the taught concept. Strategies of evaluating numerical expressions may help to detect whether a learner has knowledge on magnitude of operands and order of evaluation on operators in an expression. The deficiency of this knowledge accounted for twenty percents of errors detected in this study. Categories 6 and 7 accounted for the last twenty percent of learners' work that might not be understandable or remain unfinished. Those 10 percent of learners' work could not be identified for any error pattern reflected that even human teachers might be unable to understand open-ended pieces of work like evaluating mathematics expressions.

4.2 Knowledge of Learner on Error Patterns

This section will report on knowledge of learners with problems in working with fractions on addition and subtraction. After careful analysing error patterns of learners in evaluating and solving simple fraction addition and subtraction problems, two categories were summarized: (1) concrete error pattern; and (2) vague idea on working with fractions. The first category includes some concrete error patterns that can be abstracted into mal rules. The second category contains error patterns that cannot be easily summarized into mal rules but reflect vague ideas and incomplete working procedures of learners. One of the most famous mal rules on fraction addition can be named as "Add numerators and add denominators". Learner with poor knowledge on fraction addition will adopt knowledge of arithmetic addition by adding the numerators of fractions in the fraction expression to give the numerator of the resultant fraction and similarly adding the denominators of fractions to give the denominator of the resultant fraction. There were four learners committing this type of error in this study. This rule might explain 3% of the errors. The second category of error pattern to be analysed involves high-level abstraction. The group of learners in this category showed no concrete error patterns. However, the pattern illustrated that these learners have some vague ideas of doing fraction addition and subtraction. Examples were illustrated in table 2.

Table 2 Vague ideas for evaluating fraction addition and subtraction expressions

	Error 1	Error 2
Learner 1 (3 score)	$\frac{3}{8}+\frac{1}{6}=\frac{9}{18}+\frac{9}{18}=\frac{18}{18}=1$	
Learner 2 (6 score)	$\frac{1}{2}+\frac{1}{3}=\frac{3}{10}-\frac{2}{10}=\frac{5}{10}=\frac{1}{2}$	
Learner 3 (0 score)	$\frac{1}{2}+\frac{1}{3}=\frac{1\times3}{2\times3}+\frac{1\times3}{3\times1}=\frac{3}{6}+\frac{2}{6}=\frac{6}{5}$	$\frac{3}{8}+\frac{1}{6}=\frac{3\times6}{8\times6}+\frac{1\times6}{3\times6}=\frac{54}{18}+\frac{3}{18}=\frac{12}{18}=\frac{6}{5}$

These erroneous presentations reflected that learners did have vague ideas about the working procedures on fraction addition. They need assistance to organize the disconnected nodes into a semantic net. Result of the studies indicated that some error patterns could be represented by mal rules. However, there were even more that cannot. An alternate method of studying error patterns of learners is to understand their work.

Identify Careless Mistake

The learner model of this study reflected that twenty percent of errors were derived from calculation or careless mistakes. Careless mistakes in this study mean transcription errors or simple computational mistakes form one step to another. The feedback and guidance will be different if an error is identified as a careless one. An IILE should handle not only problems generated from subject matters but also general problems of learner like careless mistake. An authentic guidance should provide not only advices or actions that can assist learners to formulate conceptual understanding of the subject domain but also offer help to learners derived from general problems such as careless mistakes. An IILE should attempt to distinguish careless mistake from other error patterns like human teachers.

Scaffolding Support

The forty percent of errors derived from inadequate knowledge of learners reflected that only immediate feedback may not help learner much and thus authentic guidance should be considered for facilitating conceptual understanding. A productive learning support should be an arrangement of a sequence of situations for facilitating knowledge construction [12]. The role of a mathematics-learning environment will be to help learners to learn, especially those fundamental concepts in mathematics, but not to replace mathematics learning in the conventional manner. Therefore it is fundamental for such kind of learning environment to provide scaffolding support to learner when assistance is needed. Support should gradually withdraw so that learner can stand on its own after leaving the system. Therefore a fraction IILE should be designed like a blank sheet for learner to work with fraction. Feedback and guidance are only provided when it is needed. On the other hand, learner working in the IILE who does not need support will not notice the IILE in behind.

5 CONCLUSION

Studying the learning model of learners may assist IILE designers to formulate design principles and obtain details for understanding learning states of learners. The learner model of this study modelled behaviour of learners in two aspects: error category and error patterns. Nine categories of error were identified. Forty percent of errors were derived from inadequate knowledge of learners on subject matters. Twenty percent could be explained by careless mistakes. Twenty percent involved general mathematics knowledge. The final twenty percent of erroneous work were difficult to be classified or work was not completed. Learner model of the study reflected that some error patterns could be represented by mal rules. However, there were even more that cannot. An alternate method of studying error patterns of learners is to understand their work. Result of the study indicated that IILE needed to apply a strategy to identify careless mistake so that appropriate guidance to learners can be provided. The forty percent of errors derived from inadequate knowledge of learners reflected that only immediate feedback may not help much and thus authentic guidance should be considered for facilitating conceptual understanding. A productive scaffolding support should be an arrangement of situations for facilitating knowledge construction. The future work of the study is to design ways and means to understand work of students, to devise strategy to distinguish careless mistake from other error patters, and to plan scenarios for assisting learners to learn by exploration in an IILE.

References

[1] Vosniadou, S., Corte, E.D., Glaser, R., & Mandl, H. (Eds.), "International perspectives on the design of technology-supported learning environments", Hillsdale, NJ: Lawrence Erlbaum, (1996).

[2] Duffy, T., & Jonassen, D. (Eds.), "Constructivism and the technology of instruction: A conversation", Hillsdale, NJ: Lawrence Erlbaum, (1992).

[3] Chan, T.W., "An introduction to intelligent computer assisted learning systems research and development", National Science Council Monthly, Vol. 23, No. 5, pp. 456-468, (1995).

[4] Akpinar, A., & Hartley, J.R., "Designing interactive learning environments", Journal of Computer Assisted Learning, Vol. 12, pp. 33-46, (1996).

[5] Wittmann, E.Ch., "Mathematics education as a 'design science' ", Mathematics education as a research domain: A search for identity, Dordrecht: Kluwer Academic Publishers, pp. 87-103, (1998).

[6] Berliner, D.C., "In pursuit of the expert pedagogue. Educational Researcher", Vol. 15, No. 7, pp. 5-13, (1986).

[7] Livingston, C., & Borko, H., "Expert-novice differences in teaching: A cognitive analysis and implications for teacher education", Journal of Teacher Education, Vol. 40, No. 4, pp. 36-42, (1989).

[8] Shulman, L.S., "Those who understand: Knowledge growth in teaching", Educational Researcher, Vol. 15, No. 2, pp. 4-14, (1986).

[9] Cheng, C.C., "Proactive guidance in computer-assisted language learning", Technical Report, LLL-T-24-95. Language Learning Laboratory, College of Liberal Arts and Sciences, University of Illinois, (1995).

[10] Behr, M.J., Harel, G., Post, T. & Lesh, R., "Rational number, ratio and proportion", Handbook of research on mathematics teaching and learning, New York: Macmillan, pp. 296-333, (1993).

[11] Streefland L., "Fractions in realistic mathematics education: A paradigm of developmental research", Dordrecht: Kluwer Academic Publishers, (1991).

[12] Ohlsson, S., "Knowledge requirements for teaching: The case of fractions", Teaching knowledge and intelligent tutoring, Norwood, NJ: Ablex Publishing Corporation, pp. 25-59, (1991).

Towards a Meta-Knowledge Agent: Creating the context for thoughtful instructional systems

Elspeth McKay
RMIT University, RMIT BUSINESS, School of Business Information Technology, GPO Box 2476V, Melbourne, Victoria 3001, Australia, elspeth@rmit.edu.au

This paper describes a creative approach to electronic courseware authoring. Many online learning systems adopt a generic framework in which cognitive modelling is difficult to achieve. A new CBT package called Cogniware is proposed to bridge this gap by providing a novice-learner with a dynamic instructional device designed to deliver an inclusive learning context. Learners are given the opportunity by this intelligent courseware to identify their cognitive style before embarking on the instructional material. Cogniware will use research findings on the interactive effect of cognitive style and instructional format on the acquisition of complex abstract programming concepts, involving spatial relations and logical reasoning [10], to direct the novice-learner to the instructional format that will best suit their cognitive style. Cogniware will be of interest to educators, cognitive psychologists, communications engineers and computer scientists specialising in computer-human interactions.

Keywords: **Creative learning, educational agent, instructional design, interactive learning environments**

1 Introduction

Reliable mechanisms for courseware design, which provide beneficial flow-ons from research for the training and development sectors [10] are now available. Picking out the important instructional variables (learner's spatial ability, and method of delivery) for some types of instructional outcomes, progresses our ability to provide instructional environments for a broader range of novice-learners. These advancements give the learner a choice of information-transfer-agent, instructional format and instructional event conditions. Too often novice-learners are left to stumble their way through instructional material. We now have the means to deliver customised learning environments. Generic instructional formats often provide too much information, or too little. The meta-knowledge relating to an individual's likely perception of instructional strategies brings our courseware construction into the realm of being truly thoughtful instructional systems' development. In the past, there has been a traditional view that learners adopt a generic approach to make the learning of new abstract concepts meaningful. For instance, the intellectual skill associated with absorbing concepts should be included with the verbal information conveyed during instruction [20]. Cognitive processes involved in learning concepts, are generalization and discrimination [11]. For that reason, individuals generalise from a particular response to learning, to their overall learning experience. Learners look for common attributes that new concepts share with previously encountered ones [11]. However, while still assuming a generic learner cognitive profile, there is now some evidence relating to how an individual's initial mental construct might take the form of a graphical image [5]. That image could serve as a device for mental recognition if the actual object has been seen earlier. Furthermore, mental constructs include the perceptible and non-perceptible attributes of the concept and the cultural meaning given to the name of that concept.

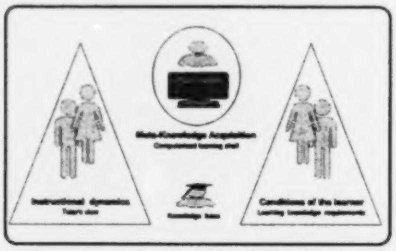
However, there are few examples of research that make a connection between learning abstract computer-programming concepts and graphical-representation as an instructional strategy (see [8]; [9] & [10]. A colour coding process to trace programming logic flow has been devised by Neufeld, Kusalik & Dobrohoczki [13]; and an interactive system, which traces the hidden activities of a computer-programming interpreter has been developed by Smith [18]. Courseware authoring involves the instructional designer in a complex pedagogical process. First, there must be some understanding of how learners deal with the learning content. Next, is the recognition of the interactive effect of an individual's knowledge processing and cognitive style. Finally, the designer needs to be aware of how the dynamics of the meta-knowledge processing (see Figure:1) impact on intelligent tutoring tools.

2 Dynamics of authoring an intelligent tutoring tool

The McKay [10] research has clearly identified the complexity of the meta-knowledge environment, and has outlined prospects for a customised learning shell. Progress is thus possible in linking research outcomes to actual learning contexts. The advent of computerized courseware dictates a need for innovative instructional strategies to articulate the visual (pictorial) approach to instruction. However, as this work has shown: not all individuals will cope effectively with a *graphical environment*

However, the observed interactive effect of the cognitive style construct [16] and instructional strategy, may be unique to the acquisition of programming concepts. Therefore, researchers/trainers will need to run an extensive pilot study programme to identify the interactive effects within their specific learning domain. In addition, the instructional material does not need to be limited to a textual/graphical comparison, but could be applied to any two or more instructional treatments of any kind. For instance, a structured versus exploratory strategy. Consequently, a special effort is required to reduce the measurable tension between the instructional mechanism (or dynamics of the tutor's view of the topic) and the actual instructional outcomes (or dynamics of the novice-learner's requirement for specific types of knowledge context). Figure:1 shows the interplay between learning and instruction.

Figure 1: Learning Process Dynamics



The Sternberg [19] approach was to concentrate on the basic information processes in analogical reasoning; while Dreyfus & Dreyfus [2] described stages of skill acquisition as five steps from novice to an expert: novice, advanced beginner, competence, proficiency, and expertise. Be that as it may, it was the sequencing of instruction that reflected the beneficial nature of meaningfulness to the act of learning [7]. Therefore, careful consideration needs to be given to the logical sequencing of instructional events to ensure participants are able to progress through the Dreyfus & Dreyfus skill acquisition steps. Intelligent tutoring systems seek to emulate the learning process, providing a novice-learner with a free fall approach to the pedagogy, or a feeling of being lost in hyperspace [4]. Many of the novice's failed attempts to construct the required domain knowledge are alleviated, when the courseware provides advance notification of the instructional content to promote the intended pedagogic framework. Thus the connection can be made between an individual's prior domain knowledge and their internal representation (Figure:1). This instructional device is called an advance organizer. It occasionally makes learning meaningful by relating new knowledge in a parallel fashion, to what is already known outside the content area [15].

3 Taking a multi-sensory approach

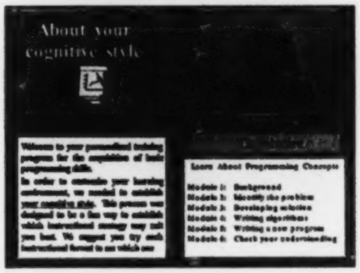
BEST COPY AVAILABLE

Multi-sensory instruction can improve a student's capacity to learn effectively [1]. This instructional approach maximises the skills brought to the learning task, while minimising the experiences where their ineptitudes are emphasised. Nevertheless, this learning process is often overlooked in the literature, in terms of making new knowledge meaningful by relating to sensory events [17], or to actions already stored in a learner's experiential database (memory). This experiential (human) database is called a sensori-motor database [6]. Accessing this human database is probably the most important method we have for making new knowledge meaningful, during the early years of cognitive development [15]. An instructional strategy can tap into the power of an individual's sensori-motor database, with an innovative textual metaphor, for explaining conditional logic flow to a novice programmer. This textual metaphor describes a common event to support a reflective approach to acquiring the programming concept of conditional logic patterns, thereby encouraging a novice-learner to access their sensori-motor database, to implement a new concept. Experiential leverage for developing the procedural knowledge is gained through providing hands-on experience with example problems. There is a relationship between cognitive level and mental energy consumption in different learning activities [14]. Reading and listening are mentally and physically exhausting with dull and poorly designed material, thereby losing the reader's interest. Furthermore, there is another relationship between cognitive level and suggestive impact, for different kinds of instructional representations [9]. Therefore, designers should be conscious of this and strive to design their learning materials (text and pictures) in the most attractive, and relevant manner possible, so that novice learners are encouraged to process the content (message) on the highest possible cognitive level.

4 Cogniware

Following the premise that a multi-sensori approach is beneficial to learning. Cogniware has been developed using the Electronic Trainer authoring tool from Mindware Creative Inc. At present it consists of a front end module to determine the learner's cognitive style (the CSA [16]), and a choice of instruction method for the acquisition of programming concepts. Cogniware is multi-sensori in the sense that the instructional strategies on offer provide the learning material in a range of alternative instructional conditions. Figure:2 depicts a typical Cogniware screen interface with three instructional formats or separate viewing areas: graphical, textual, and voice. In addition there are cueing mechanisms for guided exploration, such as: directional icons, a learning module name tag, and an advance organizer screen.

Figure 2: Towards a Meta-Knowledge Agent



Cogniware provides the background material on different modes of learning in a textual description interface, while at the same time a voice description can be heard.

4.1 Choice of instructional format

Currently, Cogniware has three types of instructional format available: graphical, textual, and voice (see Figure:2), thereby providing the learner with the format which best suits their cognitive style. However, Cogniware is also flexible enough such that a learner can over-ride the default for the chosen format. Programming metaphors are used as expository instructional strategies. In so doing, they articulate the critical attributes of the concept-to-be-learned [12].

BEST COPY AVAILABLE

4.1.1 Textual

There are a number of ways in which we can aid the comprehension of the written word. To overcome one of the central difficulties associated with text processing, Cogniware provides the reader with the best possible means to select important information from the text [3]. Hotwords are included as pedagogical cues to navigate a novice-learner through a new concept. Text should not be considered as a flat structure, where all ideas are expressed with equal importance. The Cogniware text is therefore a highly structured communication tool, in which ideas are expressed hierarchically, where certain parts of the message can receive more attention than others. As a consequence, particular display techniques enable the reader to focus on the full context of the message by selecting the important issues without being overwhelmed by poorly structured text.

4.1.2 Graphical

Graphical metaphors used by Cogniware were chosen for their recognisable and distinguishing (or salient) features, to depict each programming concept to be learned. These visual metaphors serve to elicit prior experiential knowledge, enabling the learner to recognise the distinguishing features of the new concept, and to interpret the instructional context without specific prior learning.

4.1.3 Voice

The learner can view the video interface to hear a verbal description of the programming metaphors. Advice and reassurance is also provided to ensure maximum coverage of the multi-sensori platform. Voice directions for dealing with the CBT navigation are designed to reduce the cognitive effort required in dealing with the complexities of multi-media instruction. Reminders can be seen as a useful technique to keep the novice-learner on track. It is intended that demonstration video clips will be included in future releases of Cogniware to extend the multi-sensori capability.

5 Conclusions

Cogniware represents a creative approach to electronic courseware authoring. The sound instructional design foundation upon which this courseware is built, draws on the research conducted by Merrill's ID_2 team at Utah State University, USA, and recent research by McKay & Garner [9]. The latter research provided the experimental findings to link the important work on the Cognitive Styles Construct carried out at Birmingham University, UK , by Riding [16] with the effectiveness of various instructional formats. Cogniware was authored using The Electronic Trainer providing the ideal knowledge based framework for authoring electronic courseware. Online learning systems adopting a generic framework reveal that cognitive modelling is difficult to achieve. It is proposed that Cogniware bridges this gap by providing a novice-learner with a dynamic instructional device designed to deliver an inclusive rather than exclusive learning context. At the nexus of this CBT is the ability afforded to learners to identify their cognitive style before engaging with the multi-sensori instructional devices, allowing selection of an optimal instructional format. Cogniware will be of interest to educators, cognitive psychologists, communications engineers and computer scientists specialising in computer-human interactions. Researchers can now provide a better understanding of the interactive effects of the cognitive style construct and instructional format on the acquisition of abstract concepts, involving spatial relations and logical reasoning [10].

Educational researchers are reminded to work towards ensuring their instruction works for people rather than ensuring their instruction works for the technology.

References

[1] Diana, E.M., & Webb, J.M. Using geographic maps in classrooms: The conjoint influence of individual differences and dual coding on learning facts. *Learning and Individual Differences*, **9**(3), pp.195-214, (1997).

[2] Dreyfus, H.L., & Dreyfus, S.E. *Mind Over Machine: The power of human intuition and expertise in the era of the computer,* Free Press: New York, (1986).

[3] Jonassen, D.H., Hennon, R.J., Ondrusek, A., Samouilova, M., Spaulding, K.L., Yueh, P.-H., Nouri, V., DiRocco, M., & Birdwell, D. Certainty, determinism, and predictability in theories of instructional design: Lessons from science. *Educational Technology*, **37**(January-February), pp.27-34, (1997).

[4] Kashihara, A., Uji'i, H., & Toyoda, J. Reflection support for learning in hyperspace. *Educational Technology*, **39**(5), pp.19-25, (1999).

[5] Klausmeier, H.J. Concept learning and concept teaching. *Educational Psychologist*, **27(3)**(Fal), pp.267-286, (1992).

[6] Lindsay, P.H., & Norman, D.A. *Human Information Processing: An Introduction to Psychology,* (2nd ed.) Academic Press: New York, (1977).

[7] Mayer, R.E. The sequencing of instruction and the concept of assimilation-to-schema. *Instructional Science*, **6**, pp.369-388, (1977).

[8] McKay, E. Exploring the effect of graphical metaphors on the performance of learning computer programming concepts in adult learners: A pilot study. *Educational Psychology*, **19**(4), pp.471-487, (1999).

[9] McKay, E., & Garner, B.J. *The complexities of visual learning: Measuring cognitive skills performance.* Paper presented at the 7th International Conference on Computers in Education: New human abilities for the networked society, held 4-7 Nov, Japan. G.Cumming, P.Okamoto & L.Gomez (Eds.), IOS Press: Amstredam,Vol:1, pp. 208-215, (1999).

[10] McKay, E. *Instructional Strategies Integrating the Cognitive Style Construct: A Meta-Knowledge Processing Model (Contextual components that facilitate spatial/logical task performance):* Doctoral Dissertation in Computer Science and Information Systems: An investigation of instructional strategies that facilitate the learning of complex abstract programming concepts through visual representation, Faculty of Computing and Mathematics, Deakin University, Geelong Campus, Victoria. Australia, (2000).

[11] Merrill, M.D., Tennyson, R.D., & Posey, L.O. *Teaching Concepts: An instructional design guide,* (2nd ed.) Educational Technology Publications: New Jersey, (1992).

[12] Merrill, M.D. (1994). *Instructional Design Theory,* Educational Technology Publications: New Jersey.

[13] Neufeld, E., Kusalik, J., & Dobrohoczki, M. Visual metaphors for understanding logic program execution. *Graphics Interface* '97, pp.114-120, (1997).

[14] Pettersson, R. *Using digitial image elements to produce schematic pictures.* Paper presented at the 25th Annual Conference of the International Visual Literacy Association (held October 13-17): Visual literacy in the digital age: Selected readings, New York, (1993).

[15] Reigeluth, C.M. Meaningfulness and instruction: Relating what is being learned to what a student knows. *Instructional Science*, **12**, pp.197-218, (1983).

[16] Riding, R.J. *Cognitive Styles Analysis* : Birmingham Learning and Training Technology, (1991).

[17] Rosch, E. Principles of Cognition and Categorization. In E.Rosch & B.Lloyd (Eds.), *Cognition and Categorization.* Erlbaum: New Jersey, (1978).

[18] Smith, P. *The Development and Evaluation of a Glass-Box Interpreter for Teaching Novice Programmers.* Unpublished Doctor of Philosophy, Deakin, Victoria, Australia, (1998).

[19] Sternberg, R.J. Component processes in analogical reasoning. *Psychological Review*, **84**, pp.253-378, (1977).

[20] Tessmer, M., Wilson, B., & Driscoll, M. *A new model of concept teaching and learning.* Paper presented at the Annual meeting of the American Educational Research Association (March), California, (1989).

Modeling the Tutor Using Reinforcement Learning

Roberto S. Legaspi and Raymund C. Sison
College of Computer Studies,De La Salle University, Professional Schools, Inc.
Manila, Philippines
ccsrsl@ccs.dlsu.edu.ph, ccsrcs@ccs.dlsu.edu.ph

This paper is a work-in-progress research that describes a learning framework that would allow a tutoring agent to predict future effects of different tutoring tasks over a particular class of learners. The framework would also build a model of tutoring heuristics for the agent to use and update. Lastly, the framework enhances the adaptivity of the tutoring agent as it performs on-line, in real-time learning. These can be achieved by modeling the tutor of an ITS via reinforcement learning with the temporal difference method TD(0) as the central learning procedure.

Keywords: **reinforcement learning, temporal difference method, tutor modeling**

1 Introduction

For about two decades, the Intelligent Tutoring System (ITS) paradigm has dominated the field of AI and Education [3]. One reason for this is that the ITS formed the solution to the non-adaptive Computer-Aided Instruction Systems that cannot qualitatively model the knowledge of individual students. This limitation of CAI Systems led to their inadequate reasoning capacity and prevented the immediate remediation of students. As a computer-based tutor that models the student's knowledge, an ITS effectively provides individualized instruction and remediation. Machine learning (ML) has been used for student modeling [5] and background knowledge construction [6]. However, ML can also be used for improving tutoring strategies [1].

The tutor of an ITS determines the method of instruction and remediation[1] to be applied on a given situation. Some of the complex teaching tasks that the tutor is expected to do include reviewing a previously learned topic, reviewing a related though not yet learned topic, giving a problem to solve which can be new, the same, or related to something that was given previously with equal level of difficulty, giving hints or advice, and giving explanation of the processes involved. Should any tutor provide such instruction and remediation abilities, it should at least know the next teaching task that would effectively aid its learner. This functionality can be provided by modeling the tutor using *reinforcement learning* with the *temporal difference* method TD(0) as the central learning procedure.

Modeling the tutor as a learning agent has several advantages. The first is that it makes generalization of *teaching interaction sequences*[2] easier. This means that the tutoring agent can identify the best teaching interaction sequence over a particular student. The fact that no two students are completely the same shall not be disregarded. However, the scope of generalization shall be limited. Generalization of teaching interaction sequences shall only be done over a particular classification of students where differences in learning patterns can be finite and minimal. This would require a model of tutoring heuristics to be created and constantly updated by the agent as it learns from its own experience. As much as the agent can generalize teaching interactions, the second advantage is the agent's ability to customize teaching interactions among individual learners. The last advantage is the enhanced adaptivity of the tutoring agent as it performs on-line, in real-time learning.

---

[1] Throughout this paper, instruction and remediation tasks will be collectively called as *teaching tasks.*

[2] A *teaching interaction sequence* consists of teaching tasks carried out by the tutor and responses made by the student which complete an entire tutoring session.

The topics of the paper are presented as follows. The system architecture will be discussed in section two. It is then followed in section three by a description of a reinforcement learning framework using TD(0) core learning procedure. Section four presents a general snapshot of the system exhibiting reinforcement learning and updating a tutoring model. The paper concludes with section five.

2 Architecture

The general architecture that the paper proposes as approach in modeling the tutor is illustrated in Figure 1.

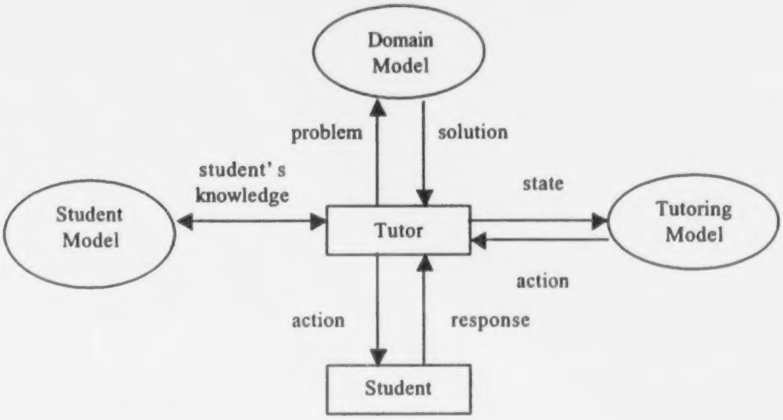


Figure 1. General Architecture to Model the Tutor

The *tutor* interacts with the *student* via the different teaching tasks (tutor' s *actions*). The student' s *response* to the tutor may range from pressing a key on the pad to giving complete solution to a problem. The *domain model* is a formal model of how the problems in the domain are to be solved. The *tutoring model* contains what the tutor has so far generated (or learned) as generalization of its teaching interaction sequences over the different student experiences. Any *state* would consist of the student' s response to a presented teaching task, what the student knows so far (the *student' s knowledge*), and the solutions to a given problem.

Once the tutor gets a clear picture of the current state, it consults the tutoring model to determine the next teaching task to be applied based on the given state. If there is no teaching task in the tutoring model that corresponds to the current state, then the tutor shall learn from experience through trial-and-error. At the end of an entire teaching interaction sequence (or tutoring session), the tutor updates the tutor model on any new sequence learned. Also at the end of an entire session, if the tutor realizes that the sequence of teaching tasks it assumed to be best for the student no longer applies, then it shifts to another sequence. For the tutor to generalize teaching interaction sequences over several students of a particular class, it shall constantly update the tutoring model.

The architecture also accounts for the peculiarities in behavior of the different students who will use it as learning happens while the student is interacting with the system. Teaching interaction sequences can be altered accordingly to individual students even on stochastic events.

3 Learning Framework

The architecture shall adopt reinforcement learning as its general learning framework with temporal difference method TD(0) as its central learning procedure. *Reinforcement learning* (RL) involves interaction between an active decision-making agent and its environment. All reinforcement learning agents have specific goals, can sense aspects of its environment, and can choose actions to influence their environment so as to achieve their goals despite the uncertainties in which they operate. Another characteristic of the RL agent is that it can use its experience to improve its performance over time. In "unknown territories", a learning agent must be able to learn from experience through trial-and-error. Furthermore, for the learning agent, on-line performance is important. Literatures have covered both generally [2] and exhaustively [6] the issues of reinforcement learning.

In reinforcement learning, an agent generates an action in some state of its environment. The way an agent maps states to corresponding actions (or the way an agent behaves), is defined by a *policy*. Each time an action is performed, a feedback in the form of a numeric -valued reinforcement or *reward* (a negative reward is interpreted as a *penalty*) is given by the environment to the agent to indicate the desirability of the resulting state. The estimate of how good it is to perform an action in some state is given by a *value function* which is defined in terms of future rewards that can be expected. The goal of the learning agent is to maximize the expected rewards. It is important to note that a value function is dependent on the particular policy being implemented. Also, rewards are factual values computed by the environment via a reward function, while the value function generates an estimate made by the agent.

Given the architecture above, reinforcement learning can be integrated as follows. The student's response (student), his current knowledge (student model), and solutions to the problem (domain model) shall comprise the environment. Depending on the prevailing policy and given a state of the environment, the tutor model predicts the action (teaching task) to take and the resultant next state and its reward. The desirability of the next state is computed using the value function. The model contains several possible next actions and their corresponding next states and next rewards but produces only one of the possibilities. The tutor takes action and waits for the student's response. This sequence of state observation, model consultation, action taking, reward assignment and value computation is repeated many times until the goal is either achieved or not. The tutor receives a positive reward when the goal is reached. Otherwise, the tutor is negatively reinforced. In the case that the model has no provision for a given state (true most especially at the initial stages of building the model), then the tutor has to learn from experience via trial-and-error while estimating value functions. At the end of an entire teaching interaction sequence, learning is achieved in two ways. First, the tutor updates the tutor model for any newly learned sequence (generalization of teaching interaction sequences) which can be used in future sessions. Secondly, the tutor can determine whether a policy worked for the student or not based on the final reward attained by the sequence. If it does not work, then the tutor shall shift to a new policy. While the tutor is now a learning agent, it is interpreted to be a tutoring agent that learns teaching interaction sequences and generalizes them.

Clearly, the learning framework requires the tutor to learn from raw experience. Another equally important requirement is that the value function, which is just an estimate, must be close to the true value to really see the efficiency of performing an action on a given state and the desirability of being in the resulting state. There must be an efficient way to update the value function estimates until convergence to the true and correct value is guaranteed. To attain these requirements, the learning framework shall adopt temporal difference (TD) method, specifically TD(0), as the central learning procedure [6]. Though TD methods learn estimates in part on the basis of other estimates, it has been found that for any given fixed policy, it guarantees convergence to the correct value. Furthermore, TD methods are naturally implemented in an on-line, fully incremental fashion.

The TD(0) algorithm can be summarized as follows. Given a state s with a value function $V(s)$ under a policy π, TD(0) initializes $V(s)$ with an estimated value and takes action a given by π for s. After a has been taken, the agent observes the reward r, the next state s', and estimates $V(s')$. Updating on $V(s)$ is done via the computation $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$ where α and γ are the learning and discount rates, respectively. Since the change in $V(s)$ to $V(s')$ now becomes a factual value, the equation brings $V(s)$ closer to its true value.

A system called ADVISOR [1] also trains an agent using RL with TD(0) as its mechanism. The agent takes its input from a model of a student population and derives a teaching policy that meets a specified educational goal. This is different from what this research would like to attain. The different teaching policies are kept by the tutor and feeds them (one at a time as needed) to the tutor model to determine the best teaching interaction sequence. At the end, what is derived is a generalization of the different teaching interaction sequences to meet the desired educational goal.

4 A General Snapshot of the System

The goal of the system' s tutor is to keep teaching time to a minimum of 30 minutes while assuring a 90% mastery of the topic for the student. Giving it a reward of +1 each time it attains the goal reinforces the tutor. But a penalty of *X-30* (where *X* is the total time to complete the teaching session) is given each time the tutor fails to attain the goal. It is assumed that the tutor will not stop (even if it exceeds the 30-minute mark) until the student gains 90% mastery of the subject matter. In such a case, the tutor receives a negative reward. The final outcome reward will tell the tutor if the policy produced a successful teaching interaction sequence or not. If it does, then the tutor continues to use that policy. Otherwise, it shifts to another policy.

Initially, the tutor model contains nothing yet. Given a policy that produced a teaching instruction sequence (review last topic, give an exercise and the student solves, explain the process, give hint and advice, give another exercise and the student solves, summarize session) that yielded the following value function estimates 0 0 0 0 0 +1. Using TD(0), the final values would be 0 0 0 0 +1 +1. The policy works in producing the correct sequence. The sequence is stored in the tutor model. As the policy prevails and the same sequence is used the next time around, then the values will become 0 0 0 +1 +1 +1. Continuously using the sequence leads to the convergence of value functions to the true values. The sequence may change (though the policy remains the same) if it no longer applies to a given state. The tutor has to learn once again from experience and come up with another training sequence and updates the model. The model contains not just the sequences that led to a success but including those that did not attain the goal.

Considering the case that the model has generalized over several situations, given a starting state, the model predicts the best teaching interaction sequence to apply on the student. Unfortunately, it did not work. The session went on to several more minutes beyond 30, and was completed with a penalty of −120. Using TD(0), the effect will cascade down through the sequence the moment it is used again. If the tutor gets into that sequence, it does not have to wait for the final outcome, using TD(0), the penalty cascaded already to some earlier state which the tutor immediately senses. At that point, the tutor abandons the sequence and changes policy.

5 Conclusion

Using reinforcement learning as the learning framework with TD(0) as the central learning procedure to model the tutor within the architecture that this paper proposes has perceived advantages over the traditional ITS. First, the tutor can provide teaching tasks that can be customized to an individual learner. Second, the system can update a tutoring model that generalizes teaching interaction sequences over several students of a particular classification. Lastly, teaching interactions are efficient as implemented in real-time.

As a future work, the learning framework can be extended to accommodate students from several different classifications.

References

[1] Beck, J., Wolf, B. & Beal, C., "ADVISOR: A machine learning architecture for intelligent tutor construction", To appear in the Proceedings of the Seventeenth National Conference on Artificial Intelligence, 2000.

[2] Kaelbing, L.P., Littman, M. & Moore, A., "Reinforcement Learning: A Survey", Journal of Artificial Intelligence, Research 4, pp. 237-285, 1996.

[3] Sandberg, J. & Andriessen, J., "Where is AI and how about Education?", Artificial Intelligence in Education, IOS Press, 1997.

[4] Sison, R., Numao, M. & Shimura, M., "Multistrategy Discovery and Detection of Novice Programmer Errors", Machine Learning, Vol. 38, pp. 157-180, 2000.

[5] Sison, R. & Shimura, M., "Student Modeling and Machine Learning", International Journal of Artificial Intelligence in Education, Vol. 9, No. 1-2, pp. 128-158, 1998.

[6] Sutton, B. & Barto, A., "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA., 1998.

Monitoring and Verifying Mathematical Proofs Formulated in a Restricted Natural Language

Peter Schmidt
Department of Computer Science, University of Bonn
Römerstr. 164, 53117 Bonn, Germany
E-mail: peter@uran.informatik.uni-bonn.de

A restricted natural language is presented which is suitable for formulating mathematical proofs in the domain of calculus. A line of a proof according to the language consists of three parts: A marking, a proof statement, and a foundation of the statement. Foundations include among others the name of a theorem, the name of a concept, or a formula manipulation operation. It is demonstrated how mathematical proofs worded in that language may be automatically monitored and checked for correctness and completeness by a computer program. For that, techniques of the fields of theorem proving and of formula manipulation are applied; the lines of the original proof are transformed into a quantifier free form and checked line by line; an internal knowledge base of concepts and theorems allows for verifying proof statements which are founded by concept definitions or theorem applications. The described methods may be used in virtual or face-to-face universities for the purpose of proof exercises by students or for the purpose of automatically checking and scoring student proofs. The approach together with a medium-grained XML representation of concepts, theorems, and proofs may form the core of a learning environment which gives students the opportunity of an intensive interactive occupation with mathematical proofs.

Keywords: **Calculus Proofs, Verifying, Restricted Natural Language**

1 Introduction

Finding and constructing mathematical proofs are standard activities of persons who study mathematics or disciplines of science. For learning purposes, it would be desirable to have an interactive software system into which students could enter a mathematical proof in the usual way utilizing the natural language and the software system would monitor and verify the student's proof or provide help if needed.

From the side of the field of mechanical theorem proving, techniques and procedures are available to automatically prove theorems or check a given proof, if the theorem or the proof are worded in a formal language like first order logic or the quantifier free clause form (see e.g. [1], [5]). The main bottleneck to reach the above mentioned goal is the difficulty of processing and correctly understanding natural language input. As a solution to the problem or as a compromise we here suggest a restricted natural language to formulate proofs. The language results from an inquiry into mathematical proofs which occur in mathematical textbooks of the domain of calculus (see e.g. [8]). We chose the domain of calculus because of the importance of calculus for the edifice of mathematics and for many practical applications and because calculus belongs to the first fields which are studied at the universities.

Secondly, we discuss how proofs utilizing that restricted language may be automatically monitored and checked for correctness and completeness by a computer program. To monitor a proof, the proof is transformed into an internal form which includes the quantifier free notations of the occurring logical expressions. A proof is checked line after line like a human would do who tries to verify a given proof. The checking for correctness of the single statements relies on the techniques of the fields of theorem proving and of formula manipulation and of their combinations. Regarding the theorem proving techniques we utilize methods which are similar to the methods of Bledsoe, Boyer and Henneman to automatically prove limit theorems ([2],[3]).

Apart from providing opportunities of doing proof exercises, the described methods may be used in virtual or face-to-face universities for the purpose of automatically checking and scoring proofs of students.

Thirdly, we shortly discuss the extension of the approach to an extensive learning environment.

2 Mathematical Theorems and Proofs in the Domain of Calculus

The *subjects of calculus* include among others limits of sequences and functions, derivations of functions, determination of properties of functions, integrals, the study of special classes of functions, and many practical applications of theoretical results.

Proof methods used in calculus are multifarious and include direct proofs using the analytical definitions of concepts like limit, continuous or differentiable (epsilon-delta notation), inductive proofs, indirect proofs or proofs by counter-examples, or direct proofs utilizing chains of inferences of already proven theorems.

A large set of proofs in the domain of calculus follows a recurrent pattern. One characteristic of those proofs is the use of analytical definitions of the main concepts to establish the proof. A further characteristic of many proofs is that they employ formula manipulation methods as a central technique to establish the proof. Proofs often consist of a construction process. Those characteristics allow for monitoring proofs without a long chain of logical deductions.

3 A Restricted Natural Language to Formulate Proofs

The restricted language to word proofs is here informally described mostly by examples so that persons who are familiar with proofs of the domain of calculus can understand the scope of the various allowable statements. The language is not supposed to be exhaustive, but the current version of the language covers a large set of calculus theorems and proofs in textbooks and in collections of exercises.

The usual structure of a natural language proof in a textbook consists of a series of statements which are substantiated by one or more foundations. The statements may have a reference to other statements of the proof. The restricted language reflects that structure by dividing a proof into proof lines. Each proof line consists of up to three parts: a marking, a proof statement, and a foundation of the proof statement. By clearly separating the three parts of a proof line from each other, the variety of natural language wording reduces to a simple and easily comprehensible structure.

3.1 The wording of a proof

Basic elements of the language. There are a series of basic elements which may occur in a proof including numbers, variable names, function names, the universal quantifier (ALL), the existential quantifier (SOME), and the logical operators of negation (NOT) and of conjunction (AND). R denotes the real numbers. *Keywords* of the language generally consist of capital letters. *Intervals* play a central role in proofs and may be designated in the usual way, e.g. [a,b] for a closed interval of the real numbers, (a,b) for an open interval, or ALL x WITH |x-a|< delta for an interval with the point a in the middle of it. *Partitions of intervals* are often used in various contexts. They usually define end points and a list of intermediate points and fix the length or a maximum length of the resulting part intervals (see an example below). *Iterations* may be used in the usual way, e.g. i=0,...,n or j=1,2, ... to denote a finite or infinite sequence.

Proof statements. The current version of the language comprises the following proof statements which are described in the next paragraphs:

(1) Assignment statements. Assignment statements allow for defining new variables or functions. An assignment statement starts with the keyword LET. Examples are
LET delta = min(delta1,delta2),
where min denotes the minimum function and delta1 and delta2 are earlier defined variables, or
LET h(x) = f(x)+g(x) ALL x IN [a,b] , where the new function h(x) is defined, or
LET f : [a,b] -> R , where a function and its domains are defined.

(2) Choice statements. Choice statements describe a choice of an entity from a set of possibilities. A choice may e.g. refer to a number chosen from an interval or to a partition of an interval. A choice statement starts with the keyword CHOOSE. The format of such a statement depends on the choice situation. Simple examples are

CHOOSE eps > 0 or CHOOSE x IN [a,b].

An example which covers the choice of a partition of an interval is

CHOOSE PARTITION p OF [a,b] WITH $a=x_0 < x_1 < ... < x_n=b$ AND ($|x_i-x_{i-1}|$ < delta , i=1,...,n) ,

where [a,b] is an interval, x_i are points in the interval, and the mentioned restriction of the lengths of the intervals $[x_{i-1},x_i]$ holds.

(3) Relational statements. Relational statements, i.e. equations and inequalities, frequently occur in calculus proofs. The statements often include constraints on the appearing variables. Typical recurrent examples relate to analytical definitions of concepts and formula manipulation operations. An example which states the definition of continuity is: ALL eps > SOME delta >0 ALL x WITH |x-a| < delta: |f(x)-f(a)| < eps. Often a chain of equations and inequalities appears like ALL x IN [a,b]: |f(x)+g(x)| <= |f(x)| + |g(x)| <= M+N < INFINITY . Another simple example of a relational statement is eps/2 + eps/2 = eps , where eps is a given variable.

(4) Property statements. Property statements describe a property of an entity, e.g. the property of a function to be continuous in an interval. An example is: f IS continuous IN [a,b]. Other properties which often occur in calculus proofs are e.g. uniformly continuous, monotonously growing, or differentiable.

A series of **further statements** which often appear in a proof more or less drive or structure the proof.

(5) Proof type statements. A *proof type statement* characterizes how the proof is done, e.g. by finding a contradiction. The statement starts with the keyword PROOF TYPE and is followed by the name of a proof method from a list of proof methods, e.g.
by DIRECT, DIRECT_BY_DEFINITION, DIRECT_BY_A_CHAIN_OF_THEOREMS, INDIRECT, COUNTEREXAMPLE, SPECIALIZATION, COMPLETE_INDUCTION. The classification of the proof may be relevant regarding several aspects which are mentioned below. An example of a proof type statement is: PROOF TYPE INDIRECT.

(6) To prove statements. *To prove statements* are used to specify what must be or will be proven. There are two variants which may precede a statement: *to prove* or *sufficient to prove*. Here are examples: Let us assume that the conclusion of a theorem is: 'The function f(x) is bounded in an interval [a,b]'. Then the first line of a proof may be e.g. TO PROVE SOME m > 0 ALL x IN [a,b]: |f(x)| < m or the first line of the proof may be e.g.

SUFFICIENT TO PROVE ALL x IN [a,b]: |f(x)| < 1 . In the first case the keywords are followed by a statement which is equivalent to the conclusion of the theorem. And in the second case the keywords are followed by a statement from which the conclusion of the theorem may be inferred.

(7) Assume statements. *Assume statements* are mostly found in indirect proofs. They then state the negation of the statement of the theorem. The statement starts with the keyword ASSUME and there follows another statement. An example is ASSUME NOT [c] , where [c] denotes the marking of the conclusion of the theorem (see an example in Theorem 2 below).

(8) Contradiction statement. A *contradiction statement* states the contradiction of statements occurring in the proof. The statement starts with the keyword CONTRADICTION and its foundation contains the contradicting statements in one or the other way. An example is CONTRADICTION {[4],[6]}. The statement says that the statements marked by [4] and [6], respectively, are contradictory (see an example in Theorem 2 below).

(9) Anchor statements and **induction step statements.** *Anchor statements* and *induction step statements* serve the purpose to structure induction proofs. The statements start with the keywords ANCHOR and INDUCTION STEP, respectively. Examples are ANCHOR n = 1 and INDUCTION STEP n TO n+1.

(10) Proof finishing statement. The *proof finishing statement* consists of the keyword QED and states that the proof is assumed to be complete.

Markings. *Markings* serve the purpose to mark statements so that other parts of the proof may refer to the marked statement. The markings consist of letters and digits embraced by brackets, e.g. [A].

Foundations. A *foundation*, possibly together with other foundations, substantiates a proof statement. There are a couple of possibilities of denoting a foundation: A foundation may consist of the name of a theorem, of a formula manipulation operation, of a property of an object, or of a line number which denotes a logical line of the current proof or of the theorem. The foundation of a logical proof line is enclosed in curled brackets whereby the single foundations are enclosed in brackets and separated by commas, e.g. {[4], [5]}.

3.2 Examples of user proofs

The following examples illustrate the use of the language to formulate proofs. Note the more often occurring *double points*, e.g. one in the proof line which is marked by [2]. That double point is necessary for reasons of uniqueness to separate the prefix containing the quantified expressions from the inequality. An alternative would be to use an IF ... THEN ... statement. The foundations starting with the letters *fm* refer to formula manipulation operations, e.g. {[fm: rewriting]} in line [7]. The theorems are here not worded according to the language. A corresponding wording is necessary when the theorems and the proofs are automatically processed by a monitoring program.

Theorem 1 (Sum of continuous functions)

```
Let
  [p1] f: R->R,   g: R->R,   a in R,
  [p2] f is continuous at the point a
  [p3] f is continuous at the point a
Then
  [c] f+g is continuous at the point a
```

Proof:

```
[1] PROOF METHOD DIRECT_BY_DEFINITION
[2] TO PROVE
    ALL eps > 0 SOME delta > 0 ALL x with |x-a| < delta : |(f(x)+g(x)) - (f(a) + g(a))|< eps {[c]}
[3] CHOOSE eps > 0
[4]  SOME delta1 > 0  ALL x with |x-a| < delta1 :  |f(x) - f(a)| < eps/2  {[p2]}
[5]  SOME delta2 > 0  ALL x with |x-a| < delta2 :  |g(x) - g(a)| < eps/2  {[p3]}
[6]  LET delta = min(delta1,delta2)
[7] ALL x IN R : |(f(x)+g(x)) - (f(a) + g(a))| =  |(f(x) - f(a)) + (g(x) - g(a))|  {[fm: rewriting]}
                                             <=  |(f(x) - f(a)| + |g(x) - g(a)|    {[fm: triangle inequality]}
[8] ALL x with |x-a| < delta : |(f(x)+g(x)) - (f(a)+g(a))| <= |(f(x)-f(a)| + |g(x)-g(a)| {[7]}
                                                     <  eps/2 + eps/2  {[4], [5]}
                                                     =  eps {[fm: simplification]}
[9] QED {[2],[8]}
```

Theorem 2 (Global Monotony)

```
Let
  [p1] f: [a,b]->R  is continuous
  [p2] f  is differentiable in (a,b)
  [p3] for all x in (a,b):  f'(x) > 0
Then
  [c] f is strictly monotonously growing in [a,b].
```

Proof:

```
[A] PROOF METHOD INDIRECT
[B]  ASSUME  NOT [c]  {[A]}
[C] SOME  x1 IN [a,b], SOME x2 IN [a,b] :   x1 < x2  AND f(x1) >= f(x2)  {[B]}
[D] ( f(x2) - f(x1) ) / ( x2 - x1 )  <= 0 {[C]}
[E] SOME x0 IN (a,b) : f'(x0) =  ( f(x2) - f(x1) ) / ( x2 - x1 ) {[Mean-value theorem]}
                         > 0   {[p3]}
[F] CONTRADICTION {[D], [E]}
[G] QED
```

4 Monitoring and Checking User Proofs

A user may enter a proof of a given theorem utilizing the above described language. The natural language proof is then transformed into a quantifier free version. That version is suitable for applying techniques of theorem proving and of formula manipulation. Each step of the user proof is checked by one of several special procedures (see below). We will first discuss the quantifier free version of the above mentioned theorems. Then we will describe the special procedures in the context of checking the proof statements of Theorem 1 and of Theorem 2.

4.1 Quantifier Free Version of a Theorem and a Proof

To check a user proof the natural language proof is transformed into a quantifier free form. Generally, the known methods of the field of mechanical theorem proving apply to get a quantifier free version (see e.g. [1], [5]), but one has to take into account some particularities which result from the fact that the proof representation exceeds first order logic:
(i) The *choice statement* corresponds to a quantification. The identifier succeeding the element CHOOSE has to be treated as a universally quantified variable, if the constraint attached to the variable represents an interval. If the constraint represents an assignment, the variable corresponds to an existentially quantified variable. An example is: A statement "CHOOSE eps > 0" has to be treated as "ALL eps > 0".
(ii) The ranges (scopes) of the quantifiers are not explicitly given in the proof. They have to be determined according to the following rule: The range ends when another quantifier with the same variable name appears or with the last appearance of the variable name.

After having dealt with those exceptions one can apply the usual transformation procedures to the proof lines which contain quantifiers. The statements of the example proofs which contain quantifiers take the following forms (an 'a' or an apostrophe is here added to the markings of the original proof lines):

The quantifier free form of Theorem 1. Figure 1 essentially shows the quantifier free form of the proof of Theorem 1 according to the transformation procedure. We assume that the reader is in general familiar with that procedure and we only mention some modifications and specific aspects which relate to the example proof.

(i) According to the transformation procedure the quantified variable names must be replaced by unique names and the existentially quantified variables are replaced by Skolem functions. In the example, the variable *eps* of line [3] is renamed into *eps0*; *delta1* and *delta2* are replaced by the Skolem functions *d1(eps0)* and *d2(eps0)* which depend on *eps0*; *delta* of line [6] is renamed into *delta0* and defined as *min(d1(eps0),d2(eps0))*; the various variables *x* are not renamed here in the example because of readability.

(ii) The equations and inequalities are assigned a corresponding interval of validity. With that we follow the proceeding of Bledsoe et al. [2].

In addition to the quantifier free version, the monitoring program utilizes a table of the occurring objects, i.e. the functions, variables, constants, and their characteristic properties. We do not here mention further details.

```
[2a] |(f(x)+g(x)) -(f(a)+g(a))|< eps                     ; x IN ( a-delta , a+delta ),
[4a] |f(x) - f(a)| < eps0/2                               ; x IN ( a-d1(eps0) , a+d1(eps0) ) ,
[5a] |g(x) - g(a)| < eps0/2                               ; x IN ( a-d2(eps0) , a+d2(eps0) ) ,
[7a] |(f(x)+g(x)) -(f(a)+g(a)) |= |(f(x)-f(a))+(g(x)) -g(a)) | ; x IN R
     |(f(x)-f(a))+(g(x)) -g(a)) |<= |f(x)-f(a)|+|g(x) - g(a)|  ; x IN R
[8a] |(f(x)+g(x)) -(f(a)+g(a)) |<= |f(x)-f(a)|+|g(x)-g(a)|     ; x IN ( a-delta0 , a+delta0 )
     |f(x)-f(a)|+|g(x)-g(a)| < eps0/2 +eps0/2                  ; x IN ( a-delta0 , a+delta0 )
     eps0/2 +eps0/2 = eps0
```

Figure 1: Quantifier free version of the proof of Theorem 1

The quantifier free form of theorem 2. Figure 2 essentially shows the quantifier free form of the proof of Theorem 2. The quantities x0, x1, and x2 are existentially quantified.

```
[C'] f(x1) >= f(x2) AND x1 < x2
[D'] ( f(x2) - f(x1) ) / ( x2 - x1 ) <= 0
[E'] f'(x0) = ( f(x2) - f(x1) ) / ( x2 - x1 )
     (f(x2) - f(x1) ) / ( x2 - x1 ) > 0
```

Figure 2: Quantifier free version of the proof of Theorem 2

4.2 Checking a proof for correctness and completeness

The monitoring procedure of the user proof consists of checking one line of the proof after the other. The whole procedure of checking a proof falls into several special subprocedures which process the different kinds of proof statements. There are the following subprocedures which generally utilize the quantifier free versions of the original statements to process the original user statement:

PROCdef: checks the correspondence between a concept and its analytical definition
PROCfm: checks formula manipulation operations
PROClogic: checks logical manipulations
PROCassume: checks the different kinds of assume statements
PROCtoprove: checks whether the succeeding statement corresponds to the statement of the theorem
PROCtheorem: checks whether a theorem may be employed in a special situation
PROCcontradiction: checks contradicting statements
PROCqed: checks whether the theorem is in fact proven

We will describe some features of the procedures in the context of checking the example proofs and mention some more details which are not immediately related to the examples. It should be obvious that the subprocedures also apply to analogous proof steps of other theorems. With the description, we use the line markings of the original proofs (like [2] or [C]), and we do not additionally mention the corresponding line markings of the quantifier free versions (like [2a] or [C']), although the procedures actually utilize the transformed statements.

Checking Theorem 1.

Line [1] states the proof method as 'DIRECT_BY_DEFINITION'. That information will be used later when the 'QED' statement of line [9] occurs (see below).

Line [2] consists of a 'TO PROVE' statement and mentions the analytical definition of the continuity of the function f(x) + g(x) at the point a and as the foundation the conclusion *[c]* of the theorem. The subprocedure *PROCtoprove* uses the subprocedure *PROCdef* to verify that the user statement and the analytical definition of continuity correspond to each other. To check that statement, *PROCdef* uses an internally provided analytic definition of the concept of continuity. The user statement and the analytical definition are compared in the quantifier free form by a unification process. The user statement is regarded as correct when a unification is possible. *PROCtoprove* utilizes the foundation of the line [2] to establish the connection between the concept of continuity and the user definition. Line [2] is internally marked and used later when the 'QED' statement is processed (see below).

A 'TO PROVE' statement may also appear in a proof e.g. to state a lemma which will be used later in the proof. In that case no foundation would be needed and a connection to the conclusion of the theorem would not be established.

Statements which explicitly state the analytical definition of a concept or vice versa infer the concept from an analytical definition are frequently found in calculus proofs. They are all treated by the subprocedure *PROCdef* in a similar way.

Line [3] mentions the choice of an *eps* > 0. That statement corresponds to a universally quantified variable *ALL eps > 0.* The statement results in an entry into the table of the entities of the proof. No further operation happens.

The lines [4] and [5] reflect the analytical definitions of continuity of the functions f and g, respectively. The foundations {[p2]} and {[p3]} trigger the comparison with the definitions of the continuity of f and of g, respectively. The subprocedure *PROCdef* establishes the correctness of the user statements as in the case of line [2]. In order to deal with the *eps/2*, in contrast to the usual *eps* without any factor, a generalized version of continuity is used: *SOME M >0 ALL eps>0 SOME delta>0 ALL x WITH |x-a|<delta: |f(x)-f(a)|* < *M** eps. A suitable factor of *eps* in the middle of the proof is often the key with continuity proofs to assure a neat < *eps* without a factor when the proof is finished. The reader will know that.

Line [6] defines the variable *delta* and its value by an expression. The statement results in an entry into the table of the entities of the proof. No further operation happens.

Line [7] gives rise to an equation and an inequality. According to the mentioned foundations, the subroutine *PROCfm* uses a simplification process to check the first equation and a *triangle inequality* subprocedure to check the second relation. Formula manipulation operations play a central role with proofs in the domain of calculus, so corresponding methods need to be available.

Line [8] divides into three relations. The first inequality is an immediate consequence of [7]. *PROCfm* checks their correspondence by standardizing the inequalities and by establishing that the interval mentioned in the line [8] is contained in the interval R of [7].

The second statement resulting from [8] establishes the statements of [4] and of [5] as foundations. *PROCfm* uses *evaluation heuristics* to handle the check of the correctness.

The third relation resulting from [8] only needs simplification which is also done by *PROCfm*.

Line [9] states that the theorem is proven. In the case of a *direct proof* one expects that the conclusion of the theorem will explicitly or implicitly occur as an inference within the proof, usually at the end of the proof. The subprocedure *PROCqed* processes the proof type of the line [1] and uses the preceding 'TO PROVE' statement which was already recognized as equivalent to the statement of the theorem to check whether the relation of the line [2] is fulfilled by the statement of line [8]. Therefore *PROCqed* uses *PROCfm* and a unification process is again employed. *PROCqed* recognizes that the proof is complete.

Checking Theorem 2.
Line [A] states the proof method as 'INDIRECT'. That information will be used later when the 'QED' statement of line [G] occurs (see below).

Line [B] mentions an 'ASSUME' statement which contains a negation of the conclusion of the theorem. The subprocedure *PROCassume* recognizes that one part of the contradiction, i.e. the part referring to the conclusion of the theorem, is established.

An 'ASSUME' statement may also be used to state something which will be proven later. That corresponds to an alternative use of the 'TO PROVE' statement.

The statement of line [C] is an immediate inference of the mentioned foundation [B]. The subprocedure *PROClogic* verifies that the statement of line [C] logically follows from the logical formula NOT [c] .

The statement of line [D] is an immediate consequence of its foundation [C]. *PROCfm* uses *evaluation heuristics* to handle the check for correctness.

Line [E] divides into two relations. The first relation consists of an application of the Mean-Value Theorem. The subprocedure *PROCtheorem* proves the correctness of the line by checking whether the premises of the mentioned theorem are fulfilled. *PROCtheorem* uses an internally provided version of the theorem. The second relation is an immediate consequence of the premise [p3] and checked by *PROCfm*.

Line [F] is founded by the statements of the lines [D] and [E]. The subprocedure *PROCcontradiction* uses *PROCfm* to check the contradiction.

Line [G] states that the proof is complete. In the case of an indirect proof one expects that a contradiction occurs and that one part of the contradiction is an inference of the negated conclusion of the theorem and the other part is a valid statement which was inferred. *PROCqed* processes the proof type of the line [A] and uses the preceding 'ASSUME' and 'CONTRADICTION' statements to verify that the proof is complete.

Error handling. In a *positive* case, a user proof can be recognized as correct and complete, that means that the occurring statements can be inferred using the corresponding foundations and that the sequence of statements actually proves the conclusion of the theorem. In a *negative* case, several types of light or severe errors may occur. From the perspective of a monitoring system which checks the various proof lines there may happen three cases in connection with each proof line:
(i) *The correct case*: The monitoring program can recognize that a statement can be inferred by using the given foundations. That positive case includes the possibility that a minor error occurred which can be clarified by a dialogue between the system and the user. The list of minor errors includes syntactical errors (e.g. regarding the language or any mathematical formula) or a lacking foundation which can be completed by the system. The completion may be possible e.g. in the case that the foundation of an obvious formula manipulation operation is missing or a reference to a preceding proof line is missing.

(ii) *The error case*: The monitoring program detects e.g. a logical error, an incorrect formula manipulation transformation, an unallowed application of a theorem, a premature 'QED' statement or no 'QED' statement. In that case the system can supply a hint to the user and the user gets the opportunity to correct the error.

The feedback in the case of multiple errors in a single statement depends on the way in which the errors are interconnected. Generally, the error possibilities are multifarious. Some multiple errors can be handled one after the other, e.g. when there are two errors in a formula. The hint that the formula is not correct may make the user rectify one error, so that only one is left.

Let us consider another example: A user enters the wrong name of the theorem which he applies and the application of the theorem is also wrong. The system would try to apply the mentioned theorem and two outcomes are possible: (a) The theorem cannot be applied or (b) the theorem can be applied. In the case of (a), a hint that the theorem is not applicable could help the user to recognize that he entered a wrong theorem name. In the case of (b), the system would state the conclusion of the theorem application. The user might then also recognize that the theorem name is wrong. In those cases the double error is reduced to one error.

(iii) *The unclear case*: The monitoring program cannot decide the correctness of a proof line. Various reasons may be responsible for that. One reason is that an important foundation is missing, e.g. a reference to the theorem which was used , so that the monitoring program cannot infer the user statement. Other reasons refer to the performance of the mentioned subprocedures: They may not be able to verify a correct statement or falsify a wrong statements in certain situations. Such a case suggests to expand the monitoring program.

5 Applications and Extensions and Pragmatics

The above described approach may be utilized for different purposes by different groups in educational institutions. Students have the opportunity to occupy themselves with mathematical proofs and do exercises which may be immediately checked for correctness and completeness.

On the other hand virtual or face-to-face universities may employ such methods in automatic on-line test systems. Proofs delivered by students could be automatically checked and scored. While students construct a proof the system might give hints in the case that foundations are missing, that there are syntactical errors, that the sequence of inferences is not complete, that a statement is just wrong, or that the student is lacking an idea how to prove the theorem. Dependent upon the amount of hints or help provided the software system might decrease the score gained.

The language as it was described above does not contain a set of symbols which are frequently used in theorems and proofs, as e.g the notation for limits, sequences, sums, integrals, or the faculty function. To integrate them one may use the notations of MathML [9]. A closer look at the proofs which are found in the text books of calculus suggests that a large set of the proofs can be worded using the above outlined language when one assumes that the usual mathematical symbols are available and some more extensions are done.

The described approach of verifying proofs demands an internal knowledge base of the concepts and theorems of calculus when proof statements are founded by concept definitions or theorem applications. Such a collection will sensibly use XML as a representation language (see e.g. [6]). See an XML representation of a theorem and of a proof on the website [7]. By utilizing that knowledge base an extensive learning environment which deals with mathematical proofs may be developed. Some aspects related to getting support with finding and constructing proofs are: One may retrieve theorems having the premises which may be used with the proof. One may retrieve a list of proof ideas of the domain and discover the one which may be useful in the current context. The roughly outlined approach to a learning environment stresses the personal proof finding and proof construction activity. A different approach to a learning environment in the field of mathematical proving relies on a general, interactive theorem prover [4].

It is obvious that one has to get used to entering a proof in the restricted natural language. An adequate interface may help to reduce the cognitive overload. Another option is to further develop the language, so that the proofs may be entered in a less restricted way and look more like textbook proofs. Such proofs might then be transformed into the restricted natural language. It is clear that the students would use such a verifying system only when the advantages outweigh the disadvantages. Some advantages are the confirmation of correctness and completeness or the detection of errors and the option of getting help.

6 Conclusions

A restricted natural language to formulate mathematical proofs in the domain of calculus was presented. It was demonstrated how mathematical proofs worded in that language can be transformed into an internal representation and checked for correctness and completeness. Some educational applications were mentioned. The extension to a learning environment was roughly outlined.

Our current prototype of verifying proofs includes an interface to enter natural language proofs, some procedures of theorem proving and an own formula manipulation system. The prototype will be further developed with respect to the methods and the knowledge bases.

References

[1] Bibel, W., "Automated Theorem Proving", Vieweg, Braunschweig, Germany, (1987).

[2] Bledsoe, W.W., Boyer, R.S., Henneman, W.H., "Computer proofs of limit theorems", Artificial Intelligence 3, pp. 27-60, (1972).

[3] Bledsoe, W.W., "Non-resolution Theorem proving", Artificial Intelligence 9, pp. 1-35, (1977).

[4] Buchberger, B., Jebelean, T., Vasaru, D., "Theorema: A System for Formal Scientific Training in Natural Language Presentation", Proceedings of the Tenth World Conference on Educational Multimedia and Hypermedia (ED-MEDIA/ ED-TELECOM 98), Freiburg, Germany, June 20-25, pp. 174-179, (1998).

[5] Chang, C.-L., Lee, R.C., "Symbolic Logic and Mechanical Theorem Proving", Academic Press, NY, (1973).

[6] Goldfarb, C.F., Prescod, P., "The XML Handbook", Prentice Hall PTR, Upper Saddle River,NJ, USA, (1998).

[7] XML representation of a theorem and of a proof, http://www.cs.uni-bonn.de/~peter/ICCE2000example.html

[8] Smith, K.T., "Primer of Modern Analysis", Springer Verlag, New York, (1983).

[9] Mathematical Markup Language (MathML) 1.01 Specification, http://www.w3.org/TR/REC-MathML.

Multimedia Intelligent Tutoring System for Context-Free Grammar

Rhodora L. Reyes, Carlo Paolo T. Galvey, Marie Christine D. Gocolay,
Eden M. Ordona and Conrado del Rosario Ruiz
Software Technology Department, College of Computer Studies
De La Sane University, Manila
2401Taft Ave., Manila
Tel: (632) 5244402, (632) 5364278
Email: ccsrlr@dlsu.edu.ph

CFG-MINTS is a multimedia intelligent tutoring system that teaches context-free grammar. The tutor model of his ITS is composed of a set of teaching strategies and an algorithm that determines which teaching action to be deployed given the goals of the system and the current state of the student model. The student model uses the Constraint-Based Modeling (CBM) approach in diagnosing the learner. CBM reduces the complexity of student modeling by focusing on the difference of the student' s solution to the ideal solution only and the analysis is reduced to pattern matching. The assumption here is that there can be no correct solution of a problem that traverses a problem state, which violates the fundamental ideas, or concepts of the domain. The system also includes features for simulating the created context-free grammar to aid in teaching.

*The paper was not available by the date of printing.

Agents in a WWW System for Academic English Teaching

Alexandra Cristea and Toshio Okamoto
University of Electro-Communications, Graduate School of Information Systems
Choufu, Choufugaoka 1-5-1, Tokyo 182-8585
Tel: +81-424-43-5621; Fax: +81-424-89-6070
E-mail:alex@ai.is.uec.ac.jp

This paper describes our research on building a free, evolutionary, Internet-based, agent-based, long-distance teaching environment for academic English. Here we will describe some of the design aspects of the system prototype, focusing especially on the adaptive features and the agents of the system.

Keywords: **Distance Education, CALL, Agent Technology**

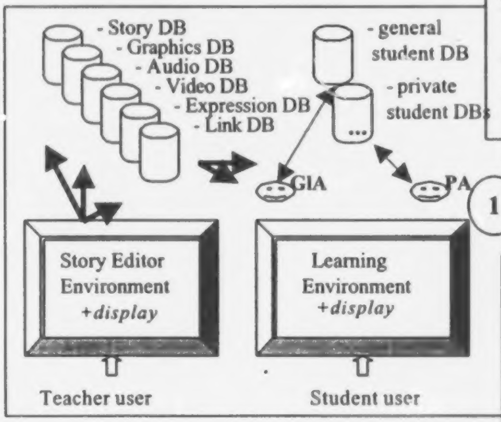
1 Introduction

As distances constantly grow smaller and the Internet links more and more remote parts of the world, English gradually becomes the lingua franca for information exchange. In the academic field, in research and development, where international cooperation is a must, English is used frequently. Academic English is International English. Although accents are more or less variable, the spoken, but mostly, the written academic language has still its rules and etiquette. Academics usually know some English and have a more or less wide English vocabulary. However, especially in Japan, but in other non-English speaking countries as well, there exists the phenomenon that, although a person can read academic papers in English, when it comes to writing a paper by oneself, or to make an academic presentation in English, serious problems appear. Therefore, we embed these necessary rules and etiquette in our teaching environment. The main aim of our system is to help academics exchange meaningful information with their peers, through a variety of information exchange ways: academic homepages, academic papers, academic presentations, etc. As far as we know, this type of English teaching system is new. Some English teaching environments on the Web appeared, but, as in [1] or [11], they have two main defects: they are not free, and/or they are not automatic, but based on real human teachers at the end of the line. Good on-line dictionaries [12], [8] and several collections of English on-line books [2] exist, but those can only act as auxiliary helpers during the English learning process. Our aim is to have a system capable to function autonomously, without human interference, as a virtual, long-distance classroom, embedding the necessary tutoring functions within a set of collaborating agents that will serve the student. The course is called 'MyEnglishTeacher', because of its evolutionary nature, of adapting over time to the needs and preferences of individual users. These needs can be expressed explicitly, or can be implicitly deduced by the system, represented by its agents. We are currently in the process of adding more AI-based intelligent adaptation capabilities. Users can find in our virtual classroom situational examples of academic life, presented as Multimedia, with Audio and/or Video presentations, Text explanations and pointers to the main patterns introduced with each lesson, exercises to test the user's understanding, moreover, adaptive correction, explanation and guidance of the user's mistakes. The general guidelines for this system were proposed by our course design researcher in [3] and elaborated by us in [6].

2 Background

Virtual environments in education and distance-learning systems are the recent trends in education worldwide. This trend is determined by the current spread of the Internet, as well as by a real demand for better, easy-to-access, and cheaper educational facilities. Therefore, universities everywhere respond to the academic demand for technological and pedagogical support in course preparation, by developing specialized software environments [5]. As bandwidths grow, the traditional text environments gradually switch to multimedia and Video-on-Demand (VOD) systems ([17]). The problems in the current language education systems, as well as the motivation of our research, as pointed out by our language specialist team member and [15], can be resumed as follows: the lack of learning activities for checking learners' constructive understanding (requiring the learner not only to memorize, but also to summarize, generate, differentiate, or predict); the lack of a variety of problem-solving tasks to motivate students to think about their reading; the learning process does not enable learners to become active participants; in the current Computer Aided Language Learning (CALL) systems, learners cannot key-in the target language's sentences freely; lack of explanatory feedback (telling the user why); lack of exercises related to the learner's individual characteristics; lack of considerations about the effectiveness of different physical attributes of the presentations, on the students' learning; lack of analysis of the interaction between learner and learning environment, with special focus on assimilation and accommodation. These problems could not be solved by traditional systems, mostly due to their lack of adaptability, or in other words, intelligence. In [19], it is stated: "there is the need to endow these systems with the ability to adapt and learn, that is, to self-improve their future performance". The objective of this research is to help learners achieve academic reading and writing ability. The course is intended for students whose starting English level is intermediate and upper-intermediate, who have some vocabulary of English, but not much practice in using it. The tutoring strategy used is to give the reader insight into his or her implicit or explicit learning strategies. The methodology applied is the communicative teaching approach, allowing communication and interaction between student and tutoring system, via agents. The interactive reading strategies applied and yet to apply include bottom-up theory, top-down theory, and schemata theory. The topics and stories used are mainly passages from textbooks, journals, reference works, conference proceedings, and academic papers, in other words, real-life academic products.

3 System features and modules



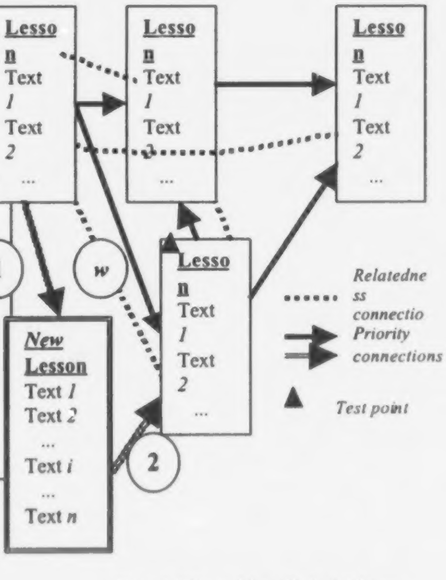


Fig. 1: The system modules and their interaction

Fig. 2: The subject link database

The system offers two interfaces, one for the teacher/tutor user, for course-authoring purposes, and the other one for the student user, who is supposed to learn. The information exchange from tutor to system contains input of lessons, texts, links between them, etc., but also asking for help in editing. The data from the tutor is stored in six different structured databases, including a library of expressions that appear in the text, a VOD database, a background image database, an audio database of listening examples, a full text database and a link database. The information exchange with the student is more complex. It contains usage of the presented materials, implicit or explicit advice, the student's advice requests, queries, searches, gathering of data on the student by the two agents, the Global Agent (GIA) and the Personal Agent (PA). Each of these agents has its own database on the student(s). The GIA stores general features on students, and the PA stores the private features of each student. User modeling follows many patterns, and has many applications. [7] proposes a fuzzy-based, stereotype collecting user model for hypermedia navigation. [18] elaborates on the Human Plausible Theory. ([4]) provides intelligent help for determining the cause of errors in software usage.

[14] has shown how prior belief (belief bias) can influence the correctness of judgment of the human (users). Other authors, like [10] have studied the relation between achievement goals, study strategies and exam performance. A realistic user model has to take into consideration the influences a system can achieve on the user, in order to allow an easy interpretation of the current state, as well as an easy and clear implementation of the user model.

4 The Authoring System Module (Story Editor)

Our most important goal is to design a meaningful, evolutionary feedback for the user. In order to build such a system, an authoring tool is necessary for flexibility purposes: our colleagues researching the optimal material for academic English teaching should be able to add or delete freely the available resources. In a way, they are also clients/users, and should be restricted to build a courseware, which conforms to the capabilities of the system. In the following, these restrictions and their purposes are explained. These restrictions are necessary instruments for the two system agents to work with, as will be shown later in this paper.

Texts: Each video/audio recording has to have a corresponding TEXT (of dialog, etc.). For each text, it is analyzed if video is necessary, or if audio suffices, as audio requires less memory space and allows a more compact storage and a speedy retrieval. Each TEXT also has (beside of main text, etc.), the following attributes: a short title, keywords, explanation, patterns to learn, conclusion, and finally, exercises. Titles and keywords are naturally used for search and retrieval, but the explanation and conclusion files can be also used for the same purpose, as will be explained later on.

Lessons: One or more TEXTs (with video or not) make up a LESSON. Each LESSON also has (beside of texts, etc.) the following attributes: title, keywords, explanation, conclusion, combined exercises (generated automatically or not). Next, a text or a lesson will be referred as 'SUBJECT'.

Priority and Relatedness Connections: When introducing one or more subjects, the teacher has to specify the Priority Connections, i.e., to show the required learning order, with a directed graph (arrows). When there is no order, subjects will have the same priority, and build a set. The teacher (courseware author) should also add connections between related SUBJECTS, with indirect links. This means, the teacher has to add Relatedness Connections between subjects, for which no specific learning order is required, but which are related. These relations are useful, e.g., during tests: if one of the subjects is considered known, the other one should be also tested. The main differences between the priority connections and the relatedness connections is that the first ones are directional, weightless connections, whereas the latter are non-directional, weighted connections. After these priorities and links are set, the system will then automatically add more links via keyword matching, from explicit keyword files and keyword search within subjects. Priorities among the texts of a lesson are set implicitly according to the order of the texts, but can be modified, if necessary. The teacher / multimedia courseware author can decide if it is more meaningful to connect individual texts, or entire lessons, for each lesson. The way a new lesson is introduced, by asking the teacher to set at least the previous and the following lesson in the lesson priority flow, is shown in figure 2 (steps 1,2). As can be noticed from figure 2, priority connections, with no respective relatedness connection, can exist. This can happen when, e.g., common course design knowledge dictates that respective priority, but the learning contents of the lessons are quite different. These kinds of priorities are optimal student learning strategy related connections, not similar contents connections. These priorities help the system to place the current subject in the global subject map. Final priorities will be set by the system according to findings (teacher's input, keyword matching). This final result can be shown to the teacher or not, depending on the options under which the system is running. We are currently testing if it is wise to allow the teacher to have add/modify/delete rights. The final graph is used for the student, and it can be shown to the student upon request, serving as a map guide.

Numbering: SUBJECTS are numbered automatically in the order of their creation. Teachers are prohibited to use numbering. This is because otherwise, every time new material is brought, the numbering should be changed according to the new order of priorities. TEXTs are automatically numbered inside a lesson, and are referred from outside with two numbers: the LESSON number and the text number.

Test Points: The teacher should mark TEST POINTS (figure 2), at which it is necessary to pass a test in order to proceed (these tests can be at any SUBJECT level).

5 Student models and agents

The system gradually builds two evolutionary student models: a global student model (GS) and an individual student model (IS), managed by two intelligent agents: the personal agent (PA) and the global agent (GIA).

The reason for doing so is that some features, which are common to all students, can be captured in the GS. However, many studies have shown [17] that personalized environments and especially, personalized tutors, have a better chance of transferring the knowledge information from tutor to student. This is true even in the more general sense of a tutor and student, where the tutor can be man or machine, and the student likewise. In this work, we mean by agent a "computer system situated in some environment", "capable of autonomous action", "in the sense that the system should be able to act without the direct intervention of humans", "and should have control over its own actions and internal state" [13]. These agents' intelligence is expressed by the fact that each agent "is capable of flexible autonomous action in order to meet its design objectives", and that it is "responsive" (it perceives its environment), "proactive" (opportunistic, goal-directed), "social" (able to interact) [13], and of an "anticipatory" nature (having a model of itself and the environment, and the capability to pre-adapt itself according to these models) [9]. Next, the raw data stored for the two student models, the GS and IS, is presented.

The GS: The GS contains the global student features: the common mistakes; favorite pages, lessons, texts, videos, audios, grading of tests' difficulty (according to how many students do each test well or not); search patterns introduced, subjects accessed afterwards: if many IS use the same order, than they are recorded in the GS.

The IS: The IS contains the personal student features: the last page accessed; grades for all tests taken, mistakes and their frequency; if the student takes the test again and succeeds, his/her last grade is deleted, but his/her previous mistakes are collected for future tests; the order of access of texts inside each lesson; order of access of lessons (this can be guide to other students: "when another student was in your situation, he/she chose..."); frequency of accessing texts/ lessons/ videos/ audios, etc. - for guidance and current state check; search patterns introduced, subjects accessed afterwards (to link patterns with new subjects that the system didn't link before).

The PA: The role of the personal agent is to manage the information gathered on the user, and to extract from this information useful user guidance material. Each step taken by the user inside the environment is stored, and compared with both what was proposed to the user, as well as with what the user was expected to do (from the PA's point of view). The differences between previous expectation and current state are exploited, in order to be used for new guidance generation. Beside of analyzing the own user and extracting knowledge from the data on him/her, the PA is able to request information from the GIA, about, for instance, what other users chose to do in a similar situation to the current one of the PA's own user. Furthermore, the PA can contact other PA's with similar profiles (after a matchmaking process), and obtain similar information as from the GIA, only with more specificity. The PA can decide to turn to another PA if the information from the GIA is insufficient for a decision about the current support method. The PA decides, every time a user enters the system, what material should be studied during that particular session, and generates a corresponding list. Therefore, the course index is dynamic, not static. To this material, the PA will add or subtract, according to the interaction with the user during the session. According to [16], the PA is therefore an interface agent ("a computer program to provide assistance to a user dealing with a particular computer application" – in this case, a learning environment). However, the PA's job description is a little wider than this, as can be seen also in the following.

The GIA: The global agent averages information from several users, to obtain a general student model. The deductions of the global agent are bound to be non-specific. The GIA is necessary, because otherwise, the system will not profit from the fact that different users interacted with the system, and each new interaction can smoothen the path for following users. The GIA is to be referred before the PA starts looking for information from other PAs, process that can be more time-consuming. Therefore, the role of the GIA is to offer to the PAs condensed information, in an easily accessible, swiftly loadable form. From this description, it is clear that the GIA is subordinate to the PA (from the student user's point of view). The GIA cannot directly contact the student user – unless the PA explicitly requests it. If the GIA considers that its intervention is required, it still has to ask for permission from the PA. In this way, the generation of confusing advice is avoided.

From the described interactions between agents and databases, and between the agents themselves, it is clear that the agents of the system work in two ways. The first way is based on the embedded rule/knowledge systems, which try to foresee, prevent and solve conflicting situations. The second way is as evolutionary, learning objects, which can adaptively change their representation of the subject space, by creating and deleting links and changing weights. A next step in the system's agents design will be focused on adaptive problem, quiz and test generation. In short, this design is made necessary by the fact that a student, after failing to pass a test, has to be presented, after some more learning is done, with a new test, of similar difficulty and contents. As it is difficult for the teachers to generate as many tests as would be necessary for such repeated situations, this task is to be passed to the system's agents. A very important task of each of the agents is also to keep the consistency of the subject link database. The agents inform the teacher(s) if some subjects form loops (determined by the priority connections set by the teacher(s)), if some subjects become inaccessible; if a teacher is not available, they make corrections by themselves, and decide from the student (s) feedback about the appropriateness of those changes.

6 Conclusions

We have proposed in this paper an Evolutionary, Web-based, Academic English Teaching Environment, called "MyEnglishTeacher". Moreover, we have described the rationale, the design and implementation and the modules of our system: an authoring environment for the teacher user(s), which is generating the lessons, and a learning environment for the student user(s). We have further on presented each of these modules in more details. The learning environment is based on two intelligent agents, interacting with each other and the student user, in order to guide the student through a new course for academic English, which is under development in our laboratory. We have also explained in which sense our agents evolve and present intelligence. Our agents build and modify student models with the help of a double graph: a non-weighted, directional priority graph, and a weighted, non-directional, relatedness graph. In addition, we have explained how, from the authoring system courseware design requirements, we enforce the generation of structured content databases, to serve as a basis to the rule/knowledge bases, which will be used and added to by the two agents. We believe that with our system we are addressing more than one current need: the need of an English tutor for academics, which should also be easily accessible – i.e., on-line –, free, adaptive and user-friendly.

References

[1] Aspera PrivaTeacher, http://www.privateacher.com/

[2] Bartleby.com, Great books on-line, http://www.bartleby.com/

[3] Cristea et al,. (1999) "The Theoretical Framework for CALL Courseware Development", Frontiers in Artificial Intelligence and Applications Series, Eds. Cumming, G., Okamoto, T., Gomez, L., IOS Press Ohmsha, 723-726.

[4] Collins, A. and Michalski, R. (1989) "The logic of plausible reasoning: A core theory", Cognitive Science, Vol. 13, 1-49.

[5] Collis, B. (1999) "Design, Development and Implementation of a WWW-Based Course-Support System", Frontiers in Artificial Intelligence and Applications Series, Eds. Cumming, G., Okamoto, T., Gomez, L., IOS Press Ohmsha, 11-18.

[6] Cristea et al, (2000) "An Intelligent language teaching tool for the Web", Proc. of ISSEI 2000, Bergen, Norway.

[7] Di Lascio, L., Fischetti, E. and Gisolfi, A. (1999) "A Fuzzy-Based Approach to Stereotype Selection in Hypermedia", User Modeling and User-Adaptive Interaction, Vol. 9, No.4, Kluwer Academic Publishers, 285-320.

[8] Dictionary, http://www.dictionary.com/

[9] Ekdahl, B., Astor, E. and Davidsson, P. (1995) "Towards Anticipatory Agents", Woolridge, M., Jennings, N.R. (Eds.), Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence, Springer Verlag, 191-202.

[10] Elliot, A.J., McGregor, H.A. and Gable, S. (1999) "Achievement Goals, Study Strategies, and Exam Performance: A Mediator Analysis", Journal of Educational Psychology, Vol. 91, No.3, 549-563.

[11] EnglishLearner, http://www.englishlearner.com

[12] Jeffrey's Japanese-English ó English-Japanese on-line dictionary. http://enterprise.dsi.crc.ca/cgi-bin/j-e/jis/dict

[13] Jennings, N.R. and Wooldridge, M. (1998) "Applications of Intelligent Agents", Agent Technology Foundations, Applications, and Markets, Springer-Verlag.

[14] Sa, W.C., West, R.F. and Stanovich, K.E. (1999) "The Domain Specificity and Generality of Belief Bias; Searching for a Generalizable Critical Thinking Skill", Journal of Educational Psychology, Vol. 91, No.3, 497-510.

[15] Levy, M. (1997) "Computer-Assisted Language Learning", Oxford Claredon Press.

[16] Maes, P. et al. (1993) "Learning Interface Agents", Proc. of the 11th Nat. Conf. AAAI, MIT/AAAI Press.

[17] Tomek, I. (1999) "Virtual Environments in Education", Frontiers in Artificial Intelligence and Applications Series, Eds. Cumming, G., Okamoto, T., Gomez, L., IOS Press Ohmsha, 3-10.

[18] Virvou, M., Du Boulay, B. (1999) "Human Plausible Reasoning for Intelligent Help", User Modeling and User-Adaptive Interaction, Vol. 9, No.4, Kluwer Academic Publishers, 321-275.

[19] Weiss, G. And Sen, S. (Eds.) (1996) "Adaptation and Learning in Multi-Agent Systems", Springer Verlag, Lecture Notes in Artificial Intelligence, Vol. 1042.

Navigation Script for the World Wide Web

Sachio Hirokawa*, Kengo Nishino and Daisuke Nagano*****
* *Computing and Communications Center, Kyushu University,*
**E-mail: hirokawa@cc.kyushu-u.ac.jp*
** *** *Graduate School of Information Science and Electrical Engineering,*
Kyushu University
***E-mail: Kengo.Nishino@ma3.seikyou.ne.jp*
****E-mail: nagano@matu.cc.kyushu-u.ac.jp*

In the World Wide Web, there is rich material for education. We propose a language to navigate students through the educational material on WWW. Navigation script makers can describe a tour with sequential, parallel and selective controls. It supports multiple Threads where video and audio accompany a browsing window. The language is described with XML and implemented in Java. So, the system can be used as an applet and as an application.

Keywords: **Hypermedia navigation ; Web graph ; XML ; JAVA**

1 Introduction

In the World Wide Web, there is rich material for education. For example, many university teachers give the contents of their lectures as their homepages. This paper describes a system which utilizes internet resources as educational material and makes them into an organized tour. The tour is described in a script language and an interpreter program navigates students through the material on WWW. Students are navigated automatically and interactively while they browse html-files and listen to and watch continuous multimedia.

We need to collect necessary pages from web resources containing a lot of garbage in order to make lists of URLs for our educational purpose. It is important to have students understand the relation between the collected pages and grasp the whole view of the field. When they do not understand the relation, or when they forget how they arrived the page, they feel that they got ``lost in webspace". There is a proposal of using web graphs as imaginary map of WWW[3]. A web graph is a directed graph whose nodes are URLs and whose edges are links between URLs. The web graph is more intuitive than just a list of URLs. But a web graph is nothing but a static representation of WWW. There is no mechanism how to lead students with material on the graph. There is no dynamic process to navigate them. We propose a script language that describes the navigation of WWW.

Maps are useful for navigation of real world and for navigation of WWW. For example, the page of Mapion http://mapion.co.jp/ shows geographic maps of towns. Besides, ``car navigation systems" based on GPS are becoming popular. RWML[5] and NVML[6] are proposal to combine the geographic map and the information on WWW. NVML describes the driving course, distance, time and supplies messages and images for specified points. When the car passes the point, a message and a image will appear according to a signal from GPS. The main concern of these researches is in geographic maps and navigation in real world. The maps we consider are imaginary maps of internet resources. Our goal is to design a language to describe a tour of WWW and to implement an interpreter of the language.

Ariadne[4] is a system of WWW navigation. It has a browser window and a separate window of tour. A user views the map of the tour and can proceed forward, backward and can choose if there are branches on the tour. But user needs to control every step of navigation. Our system supports both interactive and automated navigation. Another feature of our system which lacks in Ariadne is the parallel navigation. In our system, while a user is watching a browser window, another navigation thread can play audio data.

WebOFDAV[1] is a visualization system of web graph. When a user is traversing a series of URLs, the system draws the local graph of visited pages. The graph changes dynamically following the user. WebOFDAV is useful to tell where we are on WWW and powerful to get rid of the problem of lost in webspace. But the graph is used only for an aid for browsing and no navigation route is provided.

The rest of paper is organized as follows. The section 2 analyses the basic feature of navigation of WWW. The section 3 describes the navigation script using XML and explains the visualization of the scripts. The section 4 introduces a virtual machine with two stacks, which enables forward and backward navigation. The section 5 summarizes the paper.

2 Navigation Script

The most important feature of the navigation system is to guide the user around web pages in specified order. Therefore, we adapt sequentiality into navigation language. And to make the contents of html-files easier to understand, we need to combine audio, video, and images together with the usual browsing window. We introduce parallelism. To increase the variation of the navigation depending on each visitor, we add selection mechanism in the language. We design the language as a structured programming language with sequential, parallel and selective controls. The basic navigation units are multimedia data specified as URLs.

We chose XML as the description language of the navigation for simplicity and extendability. As implementation language we chose Java. We use ``XML Parser for Java"[2] for XML parser,and ``JMF"[7] for multimedia data. We describe the language as the following DTD (Document Type Definition).

```
<!ELEMENT statement (simple|sequential|parallel|select)>
<!ELEMENT simple (message)>
<!ATTLIST simple kind CDATA #REQUIRED
   target_name CDATA #REQUIRED
   play_time CDATA #REQUIRED
   delay_time CDATA #REQUIRED>
<!ELEMENT message (#PCDATA)>
<!ELEMENT sequential (simple|sequential|parallel|select)*>
<!ELEMENT parallel (simple|sequential|parallel|select)*>
<!ELEMENT select (selector)+>
<!ELEMENT selector (simple|sequential|parallel|select)>
<!ATTLIST selector selectname CDATA #REQUIRED>
```

Each tag and parameters have the following meaning.

- <statement>: This tag represents the root of navigation tour. It may contain subtours as children. There are four kinds of tours,<simple>,<sequential>,<parallel> and <select>.
- <simple>: This is the basic unit of the navigation. It contains a few lines of messages to describe the contents of the web page. It has the attributes of kind, target name, play time and delay time. Target name specifies the URL of the data. The kind describes the kind of multimedia data. Play time is the duration time and delay time is the time to wait before play.
- <sequential>: It may contain subtours of the kind <simple>, <sequential >, <parallel> and <select>. Subtours are followed consecutively.
- <parallel>: It may contain subtours of the kind <simple>, <sequential>, <parallel> and <select>. Subtours are activated in parallel.
- <select>: This tag causes a pause of the system. User can choose the navigation selectively from the given subtours. Those subtours are provided as children with the tag <selector>.
- <selector>: It may contain a subtour of the kind <simple>, <sequential>, <parallel> and <select>. It has the selectname as an attribute, which is used in the select menu.

3 Navigation Window, Control Panel and Browsing Window

Fig 1 is a screenshot of the system, which has a browser window, a quicktime movie screen, the controller screen and the window of navigation script.

We chose the representation with nested boxes for the visualization of navigation script instead conventional DOM-tree of XML nodes for several reasons. The most important feature of the navigation is the flow of time. To visualize this, we draw the subtours of a sequential tour from left to right. In Fig 1, time goes horizontally from left to right. Parallel tours and visualization of selection are placed vertically. The difference is that each subtour of the selection has its name, specified with its selectname, and the order in the choice. For example, if a selection has three choices, the second subtour is displayed as ``2/3 selectname".

Visualization of navigation script is not only for static view. It has a control panel and user can go forward and backward along the navigation. When a node is displayed on the browser window, the node in the navigation window is highlighted. So, the user has always global view of the navigation.

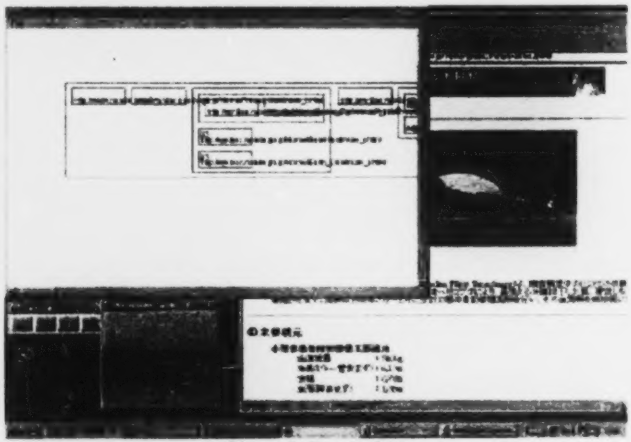


Figure 1: Screenshot

4 Interpreter of Navigation Script

Navigation is performed according to the kind of statement. Parallel statement opens a new browser window and a different thread performs the navigation in parallel.

The interpreter has two modes, the fully automatic mode and the interactive mode. Basically, the interpreter displays the specified html-files on the browser window. It displays the html-file on the screen for "play time" and changes to the next screen. When the user wants to see in detail, he can make a pause. He can go backward as well. The controller interacts with the user. The functions of the controller are "pause", "play", "forward", "backward", "rewind" and "stop". The "play" and "pause" toggles the mode. The "forward" and "backward" are for interactive mode. The browser screen moves one step in the sequential statement. This control is different to the controllers of multimedia players for the continuous media.

To realize forward/backward control in the navigation, we use two stacks of statements in the interpreter. The first stack "do"contains the list of statements to follow. The second stack "done" contains the list of statements already performed. The interpreter is realized by a transition of states depending on the top of the two stacks.

4.1 Forward Transition

Due to the limit of space, we only explain the forward transition concerning to parallel statement. If a parallel statement contains substatements, the interpreter creates n-1 threads which begin execution with ``done" stack empty and whose "do" stack contains the substatements. For example, a parallel statement "<parallel>a b c</parallel>" creates two new threads(Fig 2).

BEST COPY AVAILABLE

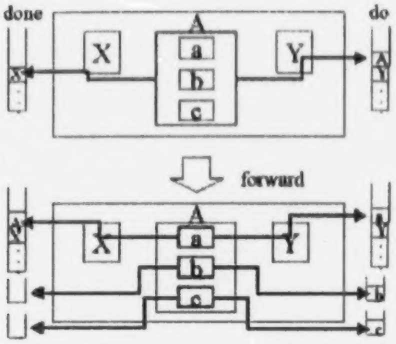


Figure 2: Forward Transition for Parallel Statement

4.2 Backward Transition

In the backward transition, the interpreter pops the statement at the top of ``done" stack and pushes it on "do" stack. If it is a sequential statement, then all the substatements are poped out of the "do" stack. A situation, where the "done" stack is empty, occurs only after a forward transition of a parallel statement. To go backward from such a situation, we need to delete such threads activated by the parallel statement.

5 Conclusions

We proposed a language for the navigation of WWW and described its implementation. The material of a navigation tour is web pages and multimedia data on WWW. The navigation script is defined as DTD of XML. Anyone can create a dynamic navigation from a static list of URLs. The language supports multimedia data and provides sequential, parallel and selective constructs of the tour.

References

[1] M. L. Huang, P. Eades, R. F. Cohen, ``WebOFDAV -navigating and visualizing the Web on-line with animated context swapping", Proceedings of 7th international WWW conference, Computer Networks and ISDN Systems, Vol. 30, no. 1-7, 638-642, (1998).

[2] IBM, ``XML Parser for Java", http://www.alphaWorks.ibm.com/tech/xml4j/(accessed on August 9,2000).

[3] D. Ikeda, T. Taguchi, S. Hirokawa, ``Developing a Knowledge Network of URLs", Proceedings of Discovery Science '99', Springer Lecture Notes in Artificial Intelligence Vol. 1721, 328-329, (1999).

[4] J. John, A.T. Jensen, K. Geronbak, ``Ariadne:a Java-based guided tour sys-tem for the World Wide Web", Proceedings of 7th international WWW conference, Computer Networks and ISDN Systems, Vol. 30, no. 1-7, 131-139, (1998).

[5] RWML Working Group, ``Road Web Markup Language (RWML) Draft Ver.0.71a (in Japanese)", http://www2.ceri.go.jp/its-win/RWML/RWML-071.html (accessed on August 9,2000), (1999).

[6] M. Sekiguchi, K. Takayama, H. Naito, Y. Maeda, H. Horai, M. Toriumi, "NaVigation Markup Language (NVML)", submission to the World WideWeb Consortium, http://www.w3.org/TR/NVML(accessed on August 9,2000), (1999).

[7] Sun Microsystems, ``Java Media Framework", http://java.sun.com/products/java-media/jmf/2.0/index.html (accessed on August 9,2000).

Organization of the introductory and motivational stage of activity in a computer tutoring system

Gennadiy Atanov and Vladimir Laktiushin
Donetsk Institute of Social Education; DISO, Donetsk, 83000, Ukraine;
E-mail: atanov@dongu.donetsk.ua

1 The activity approach in education

From the point of view of modern didactics, the final aim of instruction is not gaining knowledge but forming the way of acting being realized via skills [3]. It may be only done in the process of activity, namely learning activity. In this sense, any instructional process represents guidance, operative management of learning activity. It is management that is mechanism of teaching but not passing knowledge. Learning activity is a product of teaching because it is the aim of teaching. Knowledge is necessary, so far as the way of acting is worked out by means of operating with knowledge. On the other hand, knowledge is formed only in the process of activity []. Thus content of teaching includes subject to mastering and knowledge on which based this activity. From the point of view of organization, activity has three stages: 1) of introductory and motivation, 2) of operation and cognition, 3) of control and estimation.

An action is a unit of activity. The way of acting is a system of operations that provides solving of problems of a definite kind. The way of acting has three functional parts: (1) orientating one that provides preparation the student to activity; (2) executive one providing transformation of the objects of activity; (3) control one that provides check-up of rightness solving the problems and comparison the factual products of activity with desired ones, that is, the aim of activity [2].

Many authors of computer technologies attribute them to the ones based on activity (learning by doing) only because of work specificity with a computer but not because they realize principles of the theory of activity. In accordance with it, projecting a computer tutoring system means, first of all, projecting learning activity, not knowledge. Knowledge is projected after actions. Only on determining actions, it is possible to pick out knowledge providing formation of these actions.

Development of activity my be schematically represented in the following way: *need – motive – aim – subaims – problems – subproblems – actions – operations – product.* The introductory and motivation stage of activity, especially for learning activity, is the most important one because it is the initial stage of activity. It is called to settle questions of "lead-in" of students in activity, their adaptation to future activity, that is, questions of orienting and motivation.

In the orientating part of the way of acting, they pick out two components (Mashbits, 1988). The first one – general orienting – provides picking out those properties and qualities of the objects of activity that are essential for their transformation. The second one – orienting for the executive part – provides working out a plan of activity. Only the executive part of the way of acting providing immediate transformation of the objects is the direct product of the traditional teaching. There is the only way to do this – solving problems.

2 Organization of the introductory and motivational stage

The introductory and motivational stage plays an important psychological and didactical role in teaching in general; while using a computer, its role increases repeatedly. Nevertheless, to meet a tutoring system in which due attention would spare to this stage is a very rare thing. We organized it in tutoring systems in physics [1]. The tasks of the introductory and motivational are realization and understanding by the students: 1) aims and problems of the system; 2) physical character of processes and phenomena, as well as principles of operation of the installations that are the subjects of the system's activity; (3) knowledge necessary to reach the aim put the system. According to the theory of activity, it is operating with this knowledge that leads to forming first skills necessary for solving a particular problem and then the way of acting in aggregate.

The approach of problems that is realized in our systems is based on solving a separate problem whose complication increases that of problems being solved usually. This approach is more preferable from the point of view of activity. Firstly, it allows easily and effective organize learning activity and, secondly, it gains essentially in motivation as presupposes achievement of a practically significant aim. In many systems, this aim is even submitted in their titles, for example, "Hit the Target", "Rescue the Friends", "Render Harmless of the mine", "Determine the material". It is a very effective means to increase motivation, as the student becomes a subject of activity, the main acting person of the events expanded. Various methods of realization of this stage are used, for example, mimicking processes and phenomena, "assembling" installations from their separate parts, discussion their purposes and peculiarities of operating the installations, test tasks of the closed and open types, ones for accordance and ones for correct sequence.

Let us consider as an example systems "Internal Combustion Engine". The aim of it is determination of power and efficiency of an engine in accordance with its constructive parameters. As one can see, the title of this system does not promote increase of motivation because of the lack of the personal orientation. This is achieved by another method. A list of cars with demonstration of their outward appearance is offered to students. Students choose a car that they like and then carry out calculations for the engine of their own car.

Let us describe in what way a test task for accordance is realized in these systems. A "dumb" scheme of an installation without pointers of its component parts is shown on the screen. A list of its component part is placed next to it. Activity of students consists in the following. Separate elements of the scheme are pointed sequentially by chance, and students have to put for each element of the scheme corresponding one of the list. If the title of the pointed component part is determined correctly, another element is pointed, and so on. The determined parts acquire their numbers, and as the result, the "dumb" scheme is transformed into a "live" one. In such a way an orienting support of activity is created.

Further development of the introductory and motivation stage in the system above proceeds in the following way. The system demonstrates work of the engine during a whole cycle with replacement of the piston, opening and closing the exhaust and inlet valves, ignition of a air and gas mixture. Students may start such a demonstration several times independently. Now students see interaction of the component parts of the engine already well known to them, now they unit in their consciousness not simply mechanically but functionally reflecting physics of the processes occurring in the engine.

Subsequent deepening of orienting passes by discussion of what students have seen. It is very convenient to use the so-called active prompts with this purpose. Active prompt is built as a test task of the open type. It represents a phrase, in which a keyword is missed; this word has to be entered by students. If students do not know it, they may address to the system for help, and it will show this word on the screen. In order to keep the students' active position, the system offer the same active prompt repeatedly, and students must enter this already well known word themselves. Examples of active prompts are phrases: "The inlet valve is open when the piston goes down(wards)", "The spark springs up when the piston is at the upper extreme position" (the missed words are in italic). The main thing here consists in not completeness of these tasks but in importance of ascertaining these (and other) facts for forming the orienting base of the future activity.

The elements of the introductory and motivational stage are distributed throughout a system, their task is to prepare students to performing subsequent separate actions. If, for example, there is a necessity of using some formula, it is very convenient to remind it by a test task of the closed type. Students are offered several formulas, and they have to choose the necessary one. If students make mistakes, a short dialogue should be organized so that students could understand the nature of the mistakes. Then the task should be given again, the search of the answer becoming more sensitive. And the answer will be obtained without fail.

If the development of an action demands using exact wording (of laws, principles, theorems, definitions of concepts, and so on), it is expediently to employ a test task for the correct sequence. In the chosen wording, all the words are missed by chance (this does the system), and the task of students consists in that the words must be arranged correctly with the help of the mouse. It is a very creative and constructive work, it thrills, in the first place, because the sense appears little by little. Everyone can reach the sense even if he/she is not familiar with it at all.

References

1. Atanov, G.A., Kandrashin,G.V., Laktiushin,V.V. Computer Tutoring Systems for Science Education Based on the Activity Approach. New Media and Telematic Technologies for Education in Eastern European Countries. Eds. P.Kommers, A.Dovgiallo, V.Petrushin, P.Brusilovsky. Enshede: Tvente University Press. Pp. 75-79. (1997)
2. Gal'perin, P.Ya. Psychology of Thinking and Doctrine about Step by Step Forming Mental Actions. Moscow. (1966) (in Russian).
3. Mashbits,E. Psychological and Pedagogical Problems of Computating in Education. Moscow, Pedagogika (1988) (in Russian).

The Application of Uncertainty Reasoning for an Intelligent Tutoring System

Hown-Wen Chen*, Teran Wu and Dong-Yih Bau
Institute of Information Management, Da-Yeh University
112, Shan-Jeau Rd., Da-Tsuen, Chang-Hwa, Taiwan
Email :hwchen@aries.dyu.edu.tw

The activity of test and evaluation is an important part of Computer-Assisted Instruction systems. In most systems, 「absolutely learned」 and 「absolutely unfamiliar」are often used to represent the status of a student in learning a novice concept. However, for each target concept, there are usually more than one related sub-concepts with different degrees of importance. Thus, it is quite difficult to instruct each individual student effectively according to his learning status in those conventional systems. A hybrid technology of fuzzy theory and uncertainty reasoning are thus used in the research. The proposed intelligent tutoring system was designed to illustrate: 1.automatically tracking and analyzing the current learning status of a pupil, especially detecting the formation of learning barriers or misconceptions; 2. autonomously leading pupils to visit assisted learning path and thus proposing tutorials to make the learning of students more effectively. 3.linguistically explaining the implicit behavior of a pupil during the whole learning process. In addition, the mathematical course of teaching Pythagoream Theorem was used as the content of our test-bed. A simulation by hand and positive feedbacks from teachers of junior high schools illustrate the reasonableness and applicability of the proposed tutoring system.

Keywords: **Pythagoream Theorem, Fuzzy Logic, Uncertainty Reasoning, Intelligent Tutoring System**

1 Introduction

Researches about Intelligent Computer Aided Instruction (ICAI) have incrementally grown since 1970, for example, standard intelligent tutoring systems [1], or participants in virtual environments [2], or a virtual instructor in a training environment [3]. However, as known, the effectiveness of education would depend on the local culture. But, there are few intelligent tutoring systems focusing on Taiwanese students have been reported. CORAL [4] was designed as an interface system, without any artificial intelligence module of teachers' expertise, to provide a long-distance collative learning environment of virtual learning. As discussed in lots of tutoring systems, the most challenging issue is how to evaluate and diagnose the learning of students. Tests are a typical and popular method of evaluation. Taking the GRE as an example, people have taken the test through computers since 1992. The IBM co. and Arthur Anderson Co. have begun to work on the development of a computerized testing system. Such systems, which change the form of tests from conventional paper-to-pencil to on-line, are proliferating rapidly. For ICAI, it becomes more popular that the evaluation of pupils' learning should not be simply classified as「absolutely learned」and「absolutely unfamiliar」. In addition, ways of leading each individual pupil to enjoy an efficient learning experience is also pursued. In the research, we proposed an intelligent tutoring system which can afford the most appropriate tutorials to each pupil according to his learning status and thus can prevent pupils to trap into a misconception too long.

2 The Organization of Tutorials and Maintaining Principles

Before implementing our tutoring system, some special issues and adopted techniques must be introduced.

Those topics include the organization of tutorials, a way of representing pupil's learning status, and the detection of any formed misconception.

2.1 The Construction of a Hierarchical Concept Tree

In general, tutorials would be organized as a tree hierarchy of curriculum in the order of chapter, section, sub-section, paragraph, etc. Since learning a complicate concept must depend on the success of learning all its related sub-concepts, the kind of structure cannot be claimed to be suitable for both learners and instructors. That is, too few containment or precedence information about curriculum is available. Thus, learning concepts and related tutorials are re-arranged as a hierarchical conceptual tree of containment here. According to literatures [5] and interviews with teachers of junior high schools, the concepts related to learning Pythagoream Theorem for native pupils can be analyzed and constructed as Figure 1. In the tree, the learning of any parent conceptual node must follow after at least one of its children nodes.

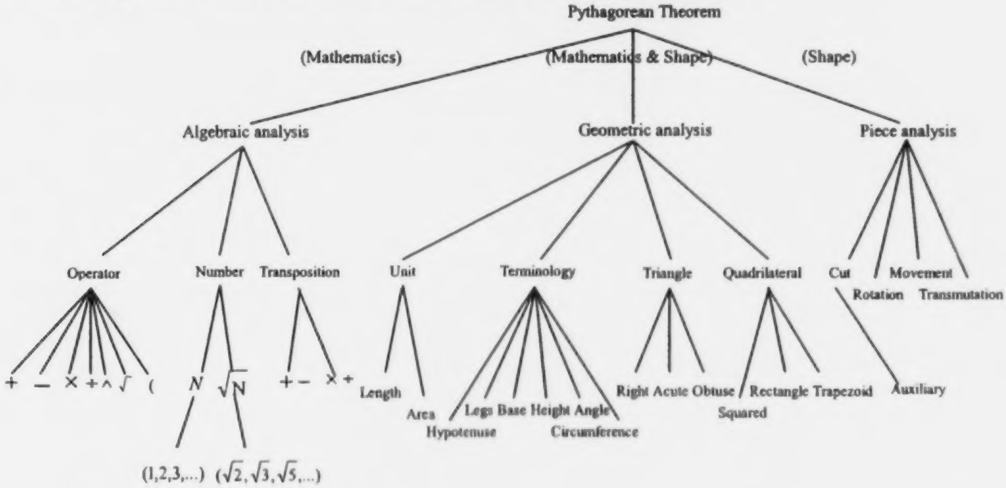

Pythagorean Theorem
(Mathematics)
(Mathematics & Shape)
(Shape)
Algebraic analysis
Geometric analysis
Piece analysis
Operator
Number
Transposition
Unit
Terminology
Triangle
Quadrilateral
Cut
Movement
Rotation
Transmutation
+ − × + ^ √ (
N √N
+ − × +
Length
Area
Legs Base Height Angle
Hypotenuse
Circumference
Right Acute Obtuse
Rectangle Trapezoid
Squared
Auxiliary
(1,2,3,...)
(√2, √3, √5,...)

Figure 1. A hierarchical concept tree of Pythagorean Theorem

2.2 The Setting of Node Weights within the Hierarchical Concept Tree

To express the corresponding degree of importance, an integer is assigned to each testing question related to individual concept [5]. However, it is still a heavy burden even for an expert to quantitatively assess the extent. Besides, the estimated grade of importance is too subjective in general. In our system, the influence of each node on learning its parent node is defined through fuzzy theory as follows:

Step 1: Some teachers in junior high schools are asked to evaluate the relevance of nodes related to their parent node in the hierarchical concept tree.

Step 2: Fuzzy theory is included to quantify teachers' opinions in the designed questionnaire obtained in step 1. Five possible values for linguistic variables are used. Note magnitudes 0.0 and 1.0 are not adopted in the memberships because of product operations and symmetry.

Step 3: Murray's or Ishihawa's Max-Min method is used to fuzzily integrate those multiple expertise. After that, a defuzzification process to evaluate the mass centroid of fuzzy numbers is applied. The weights of nodes within the hierarchical concept tree are thus settled as shown in Figure 1.

2.3 The Maintenance of Belief Parameters

To increasing the expression power of the proposed system above "absolutely known" and "absolutely unfamiliar", a belief parameter m and another updating parameter λ described in Dempster-Shafer Theorem [6] are applied here to assess the familiarity degree of a pupil to a particular learning concept within the hierarchical concept tree. To describe the meaning of the updating parameters λ and θ ($\theta=1-\lambda$), two cases must be taken in account:

Case 1: Making a correct answer

λ and θ can be used to denote the belief degree of promoting to a higher level and of staying on the same level within the conceptual hierarchy, respectively.

Case 2: Making a wrong answer

λ and θ can be used to denote the belief degree of degrading to a lower level and of staying on the same level within the conceptual hierarchy, respectively.

As to defining the updating rules of the belief parameter m, a general sub-tree structure is considered. In the tree, a node f has three children nodes labeled as a, b, c, and the interconnection links are labeled as W_{af}, W_{bf}, W_{cf}

Case 1: *Making a correct answer in the test for the conceptual node a*

A promotion within the conceptual hierarchy must be activated. The belief parameters of the two relevant nodes a and f are thus modified as

$$\begin{aligned} m_f^t &= (W_{af} \bullet \lambda) + m_f^{t-1} \\ m_a^t &= (W_{af} \bullet \theta) + m_a^{t-1} \end{aligned} \qquad \textbf{(eqn. 1)}$$

m_f^t, m_a^t : the magnitudes of belief after promotion

m_f^{t-1}, m_a^{t-1} : the magnitudes of belief before promotion

W_{af} : the weight of link between nodes **a** and f

Case 2: Making a wrong answer in the test for the conceptual node **f**

A degradation within the conceptual hierarchy must be activated. The belief parameters of the four relevant nodes, **f** and its children nodes **a**, **b**, **c**, are thus modified as

$$\begin{aligned} m_f^t &= (\theta) + m_f^{t-1} \\ m_a^t &= [\, W_{af} \bullet \lambda \bullet (1 - m_a^{t-1}) \,] + m_a^{t-1} \\ m_b^t &= [\, W_{bf} \bullet \lambda \bullet (1 - m_b^{t-1}) \,] + m_b^{t-1} \\ m_c^t &= [\, W_{cf} \bullet \lambda \bullet (1 - m_c^{t-1}) \,] + m_c^{t-1} \end{aligned} \qquad \textbf{(eqn. 2)}$$

Case 3: If a correct answer is made in the topmost conceptual node, it is impossible to promote anymore. However, the belief of the topmost conceptual node is still updated with eqn. 1.

Case 4: If a wrong answer is made in the lowest conceptual node, it is impossible to degrade and the belief of the node is updated with eqn. 2.

2.4 The Strategy of Instruction

Several principles have been applied in the proposed system:

The instruction and assessment examination would only take place in the conceptual node with the largest belief. However, all assessment tests for its children nodes with weights larger than a pre-chosen threshold must be answered correctly. If the mentioned condition is not satisfied, the focus of instruction and assessment would be transferred to one of its children nodes instead.

According to Dempster-Shapfer Theorem, the procedure of normalization must be applied after each updating of belief.

There is an implicit relationship between the magnitudes of weights and belief parameter λ. To avoid the learning process to be not in progress, according to eqn. 2, the magnitude of belief updating in any child node (a) must be larger than that of parent node (f). Thus,

$$W \bullet \lambda \bullet (1) > 1 - \lambda$$

$$\Rightarrow \lambda > \frac{1}{(1+w)} \text{ for all possible w}$$

2.5 The Analysis of Learning Traces and Detection of Misconceptions

Two kinds of traversal information would be recorded during the learning process: the weighted correct rate of answering testing questions for each conceptual node, and the traversal path of all visiting nodes.

First, the weighted correct rate can be used to indicate the current comprehension degree of a concept during the learning progresses. As known, the status near to the ending of learning should be emphasized. In other words, a pupil would be regarded as having been familiar with the concept if he can finally pass the corresponding test independent of times of previous failures. To simulate the phenomenon, three kinds of information must be kept: the number of making wrong answers W, the number of making contiguous correct answers after the last wrong answer C, and the total number of answering T. The weighted correct rate is defined as 1-W / [(T-W-C)+W+2*C], i.e., 1-W/(T+C). The interpretation of the weighted correct rate would be based on fuzzy expression in our system.

Another important issue is the way of detecting the formation of a misconception. A misconception may be caused by some blind spots of learning and thus always makes the learning process trap into a loop. A good diagnosis module of a tutoring system must have such kind of detection capability and could inform the other tutorial guidance module to show some appropriate auxiliary tutorials. If the test of each child node has passed, i.e., the learner has traversed and correctly answer all questions related to the concepts of all children nodes, the conceptual node is marked as P (Passed). If a learner cannot pass the test of a conceptual node and all its children nodes satisfy one of the following two conditions, then the learner is identified as trapping in a misconception corresponding to the conceptual node. The two conditions are <i>the child node has been marked as P; or <ii>the weighted correct rate is absolutely 1 (100%).

3 The Development and Design of Our System

Based on those described ideas, a prototype tutoring system comprising a testing and evaluation module has been developed and demonstrated. Microsoft Visual FoxPro 6.0 is used under the platform of Microsoft Windows 98. There are four modules included in our system shown in Figure 2.

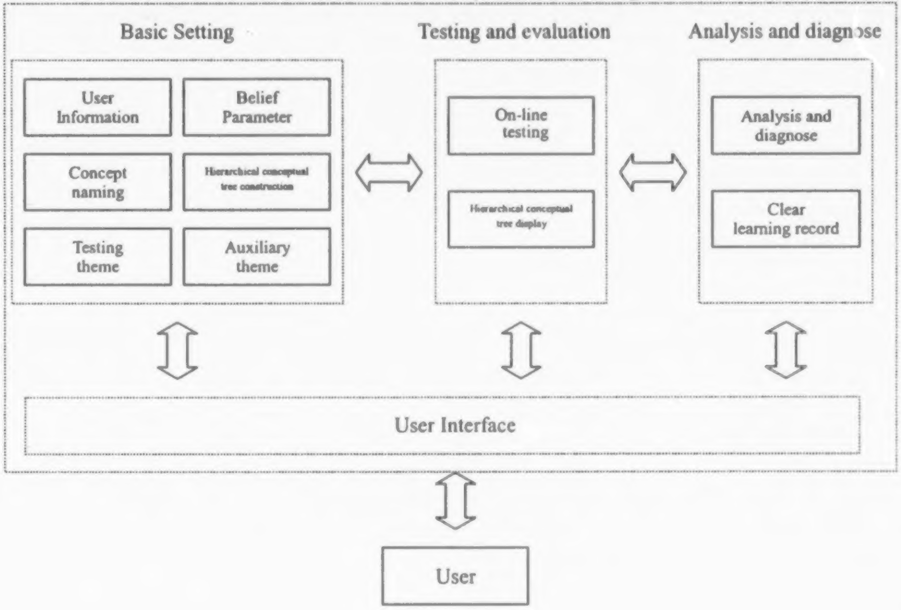

Basic Setting
Testing and evaluation
Analysis and diagnose
User Information
Belief Parameter
Concept naming
Hierarchical conceptual tree construction
Testing theme
Auxiliary theme
On-line testing
Hierarchical conceptual tree display
Analysis and diagnose
Clear learning record
User Interface
User

Figure 2. The architecture of the tutoring system

4 Conclusion and Future Work

In the research, techniques of fuzzy theory and uncertainty reasoning are applied to create a novel tutoring system. As demonstrated, the proposed tutoring system shows an excellent capability to present proper tutorials to guide pupils, precisely evaluates their learning status, and then shows auxiliary teaching materials to prevent pupils from trapping in any formed misconception. Finally, the traversal of learning would be analyzed and interpreted by fuzzy expressions.

Besides, some issues are worthy of deeper investigations through the study:

1. Some adaptive techniques of machine learning, e.g., genetic algorithm and artificial neural networks, should be applied to help instructors to automatically choose or tune parameters used in the tutoring system.
2. More applications about the proposed system should be examined to show its portability.

Acknowledgement

This study is supported by National Science Council of R.O.C. under contract number
NSC88-2520-S-212-003

References

[1] Anderson, J.R,"Cognitive modeling and intelligent tutoring", *Artificial Intelligence*. 42, pp.7-49, (1990).

[2] Pimentel, K. and Teixeira, K, "Virtual Reality: Through thenewlooking glass", Blue Ridge Summit, PA: Windcrest/McGraw-Hill, (1994).

[3] Rickel, J. and Johnson W. L, "Pedagogical agents for immersive training environments", Proc. of the 1st International Conf. on Autonomous agents, New York, (1997)..

[4] Chien Chou and Chuen-Tsai Sun, "The design and implementation of a cooperative distance learning environment: The CORAL experience", Education Technology and Media, 26, pp. 13-21, April, (1997).

[5] C. S. Hsu, H. F. Dou and G. J. Huang, "The application of concept inheritance in learning diagnosis system", Proc. of ICCAI'98, pp. 602-609, (1998).

[6] Shafer, G., "Mathematical Theory of Evidence ", Princeton University Press , Princeton , New Jersey ,(1976).

The Design and Implementation of Automatic Exercise Generator with Tagged Documents based on the Intelligence of Students:AEGIS

Tsunenori Mine **Akira Suganuma** **Takayoshi Shoudai**
Graduate School of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga-kouen, Kasuga, 816-8580 JAPAN
E-mail:{ mine@is, suga@is, shoudai@i }.kyushu-u.ac.jp

Abstract

Many Internet technologies enable us to hold lectures with Web contents and even develop new lecture methods using the technologies. This paper proposes AEGIS (Automatic Exercise Generator based on the Intelligence of Students) that generates exercises of various levels according to each student's achievement level, marks his/her answers and returns them to him/her. In order to realize this feedback mechanism, we currently restrict the question-types which are generated to the following three types: multiple-choice question, fill-the-gap question, and error-correcting question. All question-types can be generated from the same tagged document. The aim of this system is to help the students understand the lecture with exploiting preexisting electronic documents.

Keywords: Artificial Intelligence in Education, Web-Based Learning, Exercise Generator

1 Introduction

As the Internet has come into wide use, WWW environments provide lots of opportunities to various fields. In the educational domain, Web data are being exploited as useful materials. We have been developing Web-based self-teaching systems and building the tools for helping students understand their subjects[1, 2, 3, 4].

We are currently focusing on the automatic student's achievement level evaluator that generates an exercise from tagged documents, presents it to students and marks their answer automatically. We call the system AEGIS (Automatic Exercise Generator based on the Intelligence of Students)[6, 7].

Creating exercises which are suitable for students is not easy. When we try to make some exercises for them in classes, we have to take at least their achievement level into considerations. The well-considered exercises are useful not only to measure the achievement level of students but also to improve their performance. It is not easy task for any teacher to make exercises of various difficulties according to their achievement level. Besides, it is very important to mark the students' answers and return the marked results to them for keeping their learning enthusiasms. This task becomes harder in proportion to the number of the students in a class[5].

This paper discusses AEGIS, which generates the three question-types from the same tagged data. Guessing the achievement level of each student from his/her trial history, AEGIS selects the most suitable question-type and exercise for him/her according to not only his/her achievement level but also the difficulty of the tagged data. After marking his/her answer, AEGIS returns it to him/her with its explanation.

The aim of this system is to exploit pre-existing electronic documents, in particular, our on-line documents shown at our Web site (http://cl.is.kyushu-u.ac.jp/Literacy) and to help students understand their lecture whose materials are set up as Web data so that they even at home can try exercises using AEGIS through the Internet.

The rest of this paper is constructed as follows: Section 2 shows related works to discuss the difference from AEGIS. Section 3 describes question-types that AEGIS deals with, considering both view points of students(answerers) and teachers(questioners) and Section 4 describes the exercise generating process by AEGIS. Section 5 shows the overview of AEGIS.

2 Related Works

A lot of automatic quiz generators have been proposed so far. Browning et. al. proposed Tutorial Mark-up Language(TML in short) to generate questions automatically[8, 9]. TML has a couple of tags to specify a question, a multiple-choice and a message. It requires a correct answer in a multiple-choice tag to mark a student's answer to the question. Carbone et. al. proposed CADAL Quiz[10], which generates a multiple-choice quiz from a question database. After marking a student's answer, CADAL Quiz returns the result to him/her and tutors. Both of them restrict the question type only to a multiple-choice quiz. On the other hand, ClassBuilder[11] generates many kinds of quizzes and grades a student's answer. However, all of them do not mention any effect of making the difficulty level of question-type change according to the students' achievement level. In order to improve their performance and keep their enthusiasm to challenge the quiz for a long time, it is indispensable to consider their performance level for generating their exercise. This point is the difference from other systems. AEGIS makes use of pre-existing electronic documents so as to embed tags into them, generates exercises automatically with tagged documents according to students' achievment levels, and reestimates both their levels and the difficulty level of the generated question through marking their answers.

3 Question-Types

There can be several types of a question in every subject. Since our aim is to get a computer generate an exercise and mark student's answer to it, we thus restrict to the following three question-types: multiple-choice question, fill-the-gap question, and error-correcting question.

Multiple-choice question. Students choose the correct answer from a given candidate list.
Example. Complete the sentence. Choose your answer from the following list.
Data structures need to be studied _____ order to understand the algorithms.
(1) an (2) in (3) on (4) at (5) by

Fill-the-Gap question. Students try to fill in the blank of a given sentence with the correct answer without any help.
Example. Fill in the blank with the right word.
Data structures need to be studied _____ order to understand the algorithms.

Error-correcting question. Students have to find the wrong expression in a given sentence and correct it.
Example. Right or wrong? Correct the sentence if it is wrong.
Data structures need to be studied an order to understand the algorithms.

All of these question-types can be constructed from a sentence by replacing one or more consecutive words with a blank or a wrong expression. We call the region replaced *hidden region*. We note that these three question-types have different difficulties even if they are constructed from the same *hidden region*. Figure 1 shows the tagged data to be used for generating the above three types of questions.

```
⟨QUESTION SUBJECT="idioms"⟩
Data structures need to be studied ⟨DEL CAND="an,on,at,by"⟩ in ⟨/DEL⟩ order to un-
derstand the algorithms.
⟨/QUESTION⟩
```

Figure 1: The tagged data to generate three question-types shown in Section 3

Students' View Point

Every multiple-choice question has surely the correct answer in its candidate list and contains the information that leads students to the correct answer. They can therefore make their choice with confidence from the list. In the case of a fill-the-gap question, they have to fill in the blank by themselves with their convinced answer without any information about the answer. Comparing both question-types, we can say that a fill-the-gap question is more difficult than a multiple-choice one. In the case of an error-correcting question, it forces them to determine whether or not there is an error in the question sentences and to correct it if it is found. An error-correcting question gives no information leading them to its correct answer, and the wrong expression in the sentences is not clear for students. We can therefore say that an error-correcting question is the most difficult one for students among those question-types.

Teachers' View Point

Once teachers set a *hidden region*, the efforts that are required to make with the three question-types are similar. The process for making exercises is as follows: in the case of a fill-the-gap question, the teachers have nothing to do. There is no information that they have to add to the exercise paper. We can say that a fill-the-gap question is the easiest one which is made among these three question-types. In the case of an error-correcting question, teachers have to think of at least one wrong expression which can be replaced with the *hidden region*. In the case of a multiple-choice question, they have to prepare several distractors to construct a candidate list. We can say that a multiple-choice question requires more information than an error-correcting one. From their points of view, a fill-the-gap question is consequently the easiest one which is made, and an error-correcting question is easier than a multiple-choice one.

4 Automatic Exercise Generating

4.1 Exercise Generating Process

The exercise generating process from teaching documents is summarized as follows:

1. Setting a *hidden region*: teachers make clear their intention why they want to ask the question to their students, that is, they consider which of the *hidden regions* is the most suitable for their intention.
2. Selecting a paragraph or sentence(s) from teaching documents: the sentences before and after *hidden regions* are often of importance to ask their students the unique answer of the question. We call the paragraph or sentence(s) a *question region*. A *question region* may have more than one *hidden region*.
3. Constructing a candidate list: a multiple-choice question requires a couple of distractors to set up a list of answer candidates. Any distractor should be natural so as to be added to the list. This list depends on the teacher's intention.

These three steps are deeply related to the teachers' intentions. It is not easy to extract such intentions automatically from the teaching documents. AEGIS system thus deals with tagged documents that already have the information such as *hidden regions* and candidate lists.

4.2 Necessary Information for Generating Exercises

In order to embed the above three kinds of information into the teaching documents, we define the following three tags: QUESTION, DEL, and LABEL.

QUESTION surrounds a *question region*, that is, the statements between ⟨QUESTION⟩ and ⟨/QUESTION⟩ are a *question region*. In the region, there can possibly be some expressions that are related to a *hidden region*. They can be good hints to lead students to the correct answer.

SUBJECT is the unique attribute of QUESTION. Its value stands for the subject or topic of *question region*.

DEL indicates a *hidden region*, which is the word(s) or sentence(s) between ⟨DEL⟩ and ⟨/DEL⟩.

A fill-the-gap question can be generated only by replacing the *hidden region* with a blank.

CAND is one of DEL's attributes. It is used to specify a candidate list.

LABEL has an attribute NAME that specifies a dependency relation with a *hidden region*. The sentence/s surrounded by LABEL tags is/are presented as a reference for the answer of a question, which will be generated with the DEL tag whose REF's value is the same as that of the NAME of the LABEL.

⟨QUESTION SUBJECT="W_S"⟩ *question region* ⟨/QUESTION⟩

W_S ::= word or symbol, where a backslash (\) must be added just before the symbol if it is a comma (,), double quotes ("), or a backslash (\).

⟨DEL CAND="CANDIDATE" LEVEL="PAIR" GROUP="ID" REF="ID"⟩ *hidden region* ⟨/DEL⟩

CANDIDATE ::= W_S | W_S,CANDIDATE
W_S ::= word or symbol, where a backslash (\) must be added just before the symbol if it is a comma (,), double quotes ("), or a backslash (\).
PAIR ::= LOW,HIGH
LOW ::= an integer between 1 and 10
HIGH ::= an integer between 1 and 10
ID ::= keyword

⟨LABEL NAME="ID"⟩ sentences ⟨/LABEL⟩

ID ::= keyword

Figure 2: Tags for exercise generations

4.3 Necessary Information for Adjusting Difficulty Level of Question

The additional three attributes of DEL, which contain the information on the difficulty of solving the exercise, are LEVEL, GROUP, and REF. They specify the difficulty of each *hidden region*, and the connections to other *hidden region*.

LEVEL specifies the difficulty of the exercise to be generated from a *hidden region* itself. The value of this attribute is a pair of integers between 1 and 10. These integers specify the lowest and highest achievement level of the students who can try the exercise. AEGIS system determines whether or not the *hidden region* is worth being transformed into the exercise by comparing the student's achievement level from the both values of LEVEL.

GROUP specifies the dependency relation between *hidden regions* and holds the uniqueness of the correct answer. This GROUP is used to adjust the exercise level. If we want to generate more difficult exercises, all the *hidden regions* that have the same values in GROUP are replaced with blanks or wrong expressions at the same time. On the other hand, for generating easier ones, some of the *hidden regions* in the group are not transformed because those regions help students answer the question as hints.

REF specifies the dependency relation between a *hidden region* and other expressions than the *hidden region*. Both the region and expressions are specified with LABEL. If a *hidden region* is connected to an expression, the value of REF in the *hidden region* is the same as that of NAME in the expression with LABEL.

5 AEGIS system

5.1 Overview of AEGIS

The AEGIS system consists of three databases: *Exercise DB* (**EDB** in short), *User Profile DB* (**UPDB** in short) and *Level Management DB* (**LMDB** in short), and three main database managers: *Exercise Generator* (**EG** in short), *Answer Evaluator* (**AE** in short) and *Level Manager* (**LM** in short). The overview of AEGIS is shown in Fig. 3.

Teaching documents with the tags are compiled into the **EDB** and **LMDB**. All of the *question regions* are indexed sequentially and each *hidden region* is labeled with its own subindex of the index of each *question region*. The level of a *hidden region*, which is deeply related to the level of the question to be generated from the *hidden region*, is stored in the **LMDB** together with the index of the *hidden region*. The level of each *hidden region* in **LMDB** is reexamined regularly. **UPDB** keeps students' trial histories with their current achievement level.

EG and **AE** make communications with the users (students) through Web browsers after being invoked through CGI (Common Gateway Interface).

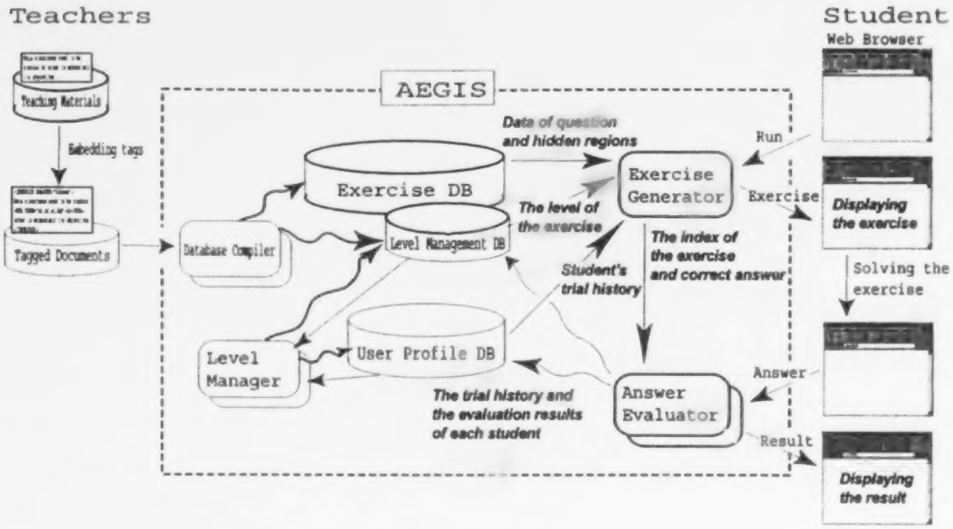

Teachers
Student
Web Browser
AEGIS
Teaching Materials
Embedding tags
Tagged Documents
Database Compiler
Exercise DB
Level Management DB
User Profile DB
Level Manager
Data of question and hidden regions
Exercise Generator
The level of the exercise
Student's trial history
The index of the exercise and correct answer
Run
Exercise
Displaying the exercise
Solving the exercise
Answer
Answer Evaluator
Result
Displaying the result
The trial history and the evaluation results of each student

Figure 3: Overview of AEGIS

5.2 Exercise Generator(EG)

The exercise request from a student invokes **EG**. The **EG** searches the most suitable *hidden region* in **EDB** with looking over both the student's profile stored in **UPDB** and the level of the *hidden region* stored in **LMDB**, and determines the question-type of the *hidden region*. As mentioned in section 3, every question level has a relation to the question-type. **EG**'s decision process of the question-type thus employs the following strategy: If the student's achievement level is closer to the lowest number in LEVEL of the *hidden region*, **EG** selects a multiple-choice question as the question-type with high probability. On the other hand, if it is closer to the highest number in the LEVEL attribute, **EG** selects an error-correcting one.

Once **EG** determines the question-type of the *hidden region*, it is not difficult to generate the question. This is because the *hidden region* represents the correct answer of the question which is generated and teachers have already given the list of distracts explicitly with CAND attribute. Now, let's see how **EG** works when it generates the three kinds of questions:

- *Multiple-choice question*: **EG** randomly constructs one possible list for the multiple choice with both the correct answer and some distracts and outputs a question, which is generated by replacing the *hidden region* with a blank, with the list.
- *Fill-the-Gap question*: **EG** outputs a question which is generated only by replacing the *hidden region* with a blank.
- *Error-correcting question*: **EG** outputs a question which is generated by replacing the *hidden region* with one of the wrong answers specified in the CAND attribute.

Figure 4 shows an example of teaching documents with the tags. It is a piece of the teaching documents in the elementary course of Computer Literacy at our university. This course is taken by all first and second year students, about 2,300 students[5]. The teacher's intention in the example document is to teach how to use multiply and divide operations. Figure 5 shows the three question-types which are generated from the document.

5.3 Answer Evaluator(AE)

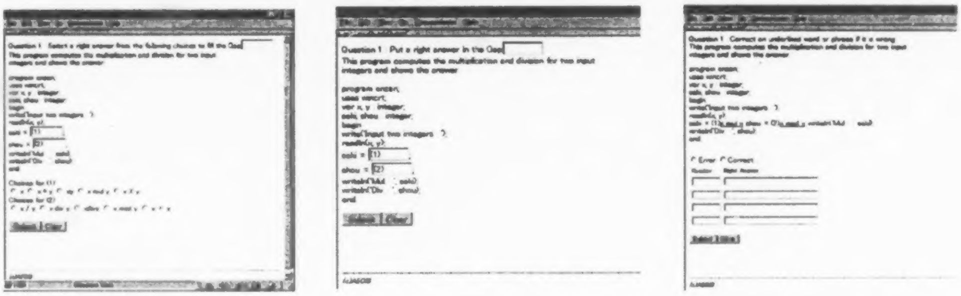
After outputting a question to the student, **EG** sends the following three kinds of information to ask **AE** to mark his/her answer: the index of a *hidden region*, the question-type, and the correct answer. After

In the previous section, we learned a program for adding two integers and showing the answer on the display. In the similar way, for all basic arithmetic operations including addition, subtraction, multiplication, and division, we can make a Pascal program in the following way.
(QUESTION SUBJECT="arithmetic operations")
This program computes the multiplication and division for two input integers and shows the answer.

```
program enzan;
var x,y:integer;
   seki,shou:integer;
begin
   write('Input two integers : ');
   readln(x,y);
   seki:=(DEL CAND="x,xy,x×y,x mul y" LEVEL="1,5")x*y(/DEL);
   shou:=(DEL CAND="x/y,x÷y,xdivy,x mod y" LEVEL="1,5")x div y(/DEL);
   writeln('Seki:',seki);
   writeln('Shou:',shou)
end.
```

(/QUESTION)
The 7th statement multiplies x by y, and the 8th statement divides x by y. We note that the answer of "div" is an integer.

Figure 4: Example of teaching documents with the tags



(a) Multiple-Choice

(b) Fill-the-Gap

(c) Error-Correcting

Figure 5: Three questions generated from the document in Figure 4

marking his/her answer by matching with the correct answer, **AE** shows him/her the marked result and stores it with the index of the *hidden region* and the question-type into the **UPDB**.

5.4 Level Manager(LM)

Although the initial value of the level of each *hidden region* is specified by teachers, it continues to move up and down according to the students' achievement levels, which will change as time goes by. The supplement manager **LM** processes their achievement levels statistically, computes the revised level of each *hidden region*, and stores it into the **LMDB**. **LM** increases the difficulty level of a question if a student whose level is greater than the level of question answers it wrongly, and decreases if a student whose level is less than the level of question answers it correctly. The new difficulty level of a question is consequently determined as shown in Fig.6.

After updating **LMDB**, **LM** updates the student's achievement level according to the difficulty levels of all questions he/she correctly answered.

Now, we show the formal definition of calculating both the achievement level of a student and the difficulty level of a question. Let $s_{i,t}$ and $q_{j,t}$ be the achievement level of student S_i and the difficulty level of question Q_j at time t respectively, where $1 \leq s_{i,t} \leq 10$,$1 \leq q_{j,t} \leq 10$. $s_{i,t}$ is recursively calculated with $q_{j,t}$ at stated periods and vice versa. They are defined as follows:

BEST COPY AVAILABLE

$$s_{i,t} = \begin{cases} 1 & \text{if } m_{s_i,t} = 0 \\ \frac{1}{m_{s_i,t}} \sum_{j=1}^{m_{s_i,t}} q_{j,t} \cdot \delta_{i,j} & \text{otherwise} \end{cases} \qquad \delta_{i,j} = \begin{cases} 1 & \text{if } S_i \text{ answered } Q_j \text{ correctly} \\ 0 & \text{otherwise} \end{cases}$$

$$q_{j,t} = \begin{cases} q_{j,t-1} + \frac{\sum_{i=1}^{m_{q_j,T}} |s_{i,\tau \in T} - q_{j,t-1}| \cdot \xi_{i,j}}{\sum_{i=1}^{m_{q_j,T}} |\xi_{i,j}|} & \text{if } \sum_{i=1}^{m_{q_j,T}} |\xi_{i,j}| \neq 0 \\ q_{j,t-1} & \text{otherwise} \end{cases}$$

$$\xi_{i,j} = \begin{cases} -1 & s_{i,\tau} \text{ is less than } q_{j,t-1} \text{ and } S_i \text{ answered } Q_j \text{ correctly} \\ 1 & s_{i,\tau} \text{ is greater than } q_{j,t-1} \text{ and } S_i \text{ answered } Q_j \text{ wrongly} \\ 0 & \text{Otherwise} \end{cases}$$

Where $m_{s_i,t}$ stands for the number of questions that S_i tried by t and τ is the latest time such that S_i tried to answer Q_j and $t-1 < \tau \leq t$. T is the set of τ. $m_{q_j,T}$ stands for the total number of students who tried Q_j in T. $q_{j,0}$, which is the initial difficulty level of the question Q_j, is given with the attribute $LEVEL$ of DEL tag by teachers.

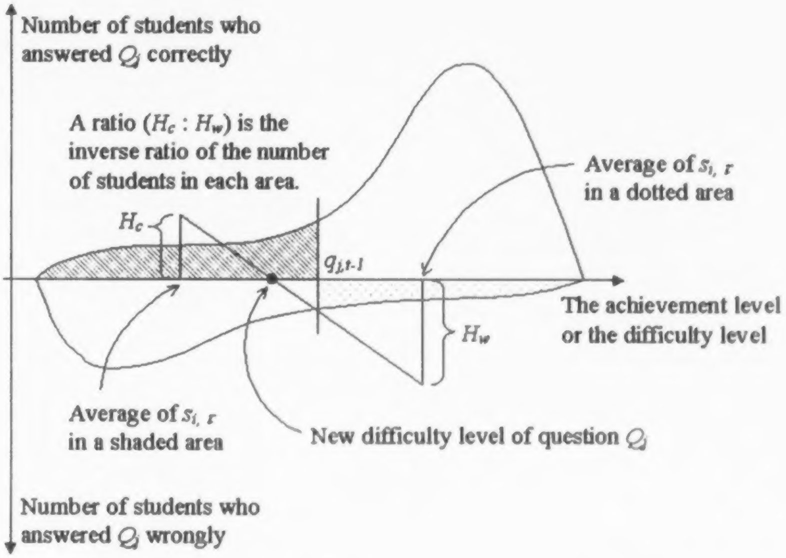

Number of students who answered Q_j correctly
A ratio ($H_c : H_w$) is the inverse ratio of the number of students in each area.
H_c
$q_{j,t-1}$
Average of $s_{i,\tau}$ in a dotted area
The achievement level or the difficulty level
H_w
Average of $s_{i,\tau}$ in a shaded area
New difficulty level of question Q_j
Number of students who answered Q_j wrongly

Figure 6: Renewing Difficulty level of Question based on Student's Achievement Level

6 Conclusions

We discussed our new Web-aided system AEGIS. The system is currently implemented in Perl scripts and CGI. We have a plan to evaluate this system by applying it to the real courses of Computer Literacy, which are taken by more than 2300 students at our university. We hope it will work fine as an educational tool for every student and help him/her to understand his/her subjects if teachers can make tags in their teaching documents. Also, we plan to implement a tagging tool and an algorithm to generate another kind of exercise that allows more than one correct answers.

Acknowledgments

We thank Dr. Teruko Mitamura at Carnegie Mellon University for her helpful comments to improve this paper.

This research was partly supported by the Grant for Special Academic Research P&P from Kyushu University and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Encouragement of Young Scientists(A), No. 11780281, 2000.

References

[1] H. Sato, T. Mine, T. Shoudai, H. Arimura, S. Hirokawa: On web visualizing how programs run for teaching 2300 students. In *Proc. Int. Conf. on Computers in Education, ICCE'97*, pages 952–954, 1997.

[2] T. Mine, D. Nagano, K. Baba, T. Shoudai, S. Hirokawa: On-web visualizing a mechanism of a single chip computer for computer literacy courses. In *Proc. Int. Conf. on Computers in Education, ICCE'98*, volume 2, pages 496–499, 1998.

[3] R. Fujimoto, A. Suganuma, T. Mine: Development of a classroom management system on the web. In *Proc. Int. Conf. on Computers in Education, ICCE'99*, volume 2, pages 756–759, 1999.

[4] R. Fujimoto, Y. Tsutsumi, A. Suganuma: Cacce: Computer aided cooperative classroom environment. In *World Conference on Educational Multimedia, Hypermedia & Telecommunications*, page 1686, 1999.

[5] S. Hirokawa, T. Miyahara, T. Mine, T. Shoudai, M. Mori: Teaching 2300 students with www – practice and experience at kyushu university. In *Proc. ERI'96*, pages 59–63, 1996.

[6] T. Shoudai, A. Suganuma, T. Mine: AEGIS: Automatic Exercise Generator with Tagged Documents based on the Intelligence of the Students, Joint Conference on Knowledge-Based Software Engineering, to appear. JCKBSE2000, Brno, Czech Republic, 2000

[7] T. Mine, T. Shoudai, A. Suganuma: Automatic Exercise Generator with Tagged Documents

Considering Learner's Performance Proceedings of WebNet2000, San Antonio, Texas, to appear as a short paper, 2000

[8] P. Browning, J. Williams, D. Brickley, H. Missou: Question Delivery over the Web using TML http://www.ilrt.bris.ac.uk/netquest/liveserver/qbanks/demos/paul/lboro/ohp1.html, CAA, 1997

[9] P. Browning: TUTORIAL MARKUP LANGUAGE - A CBA SYSTEM http://www.soton.ac.uk/~ukgec/workshop/5-cba/minutes.htm#TUTORIAL, 1998.

[10] A. Carbone, P. Schendzielorz: A Web-Based Quiz Generator for Use in Tutorials and Assessment Global J. of Engng. Educ., Vol.I, No.3, 1997 http://www.eng.monash.edu.au/usicee/gjee/vol1no3/paper20.htm.

[11] ClassBuilder GradeBook And Exam Creation Software: http://www.classbuilder.com/

Traversing the Case Graphs
A Computer Model for Developing Case-based Learning Systems

Isaac P.W. Fung & R. H. Kemp
Institute of Information Sciences and Technology
Massey University
Palmerston North, New Zealand
Fax: +64-6-3502259
P.W.Fung@massey.ac.nz; R.Kemp@massey.ac.nz

This paper presents an extended theory for representing cases in a case-based physics learning environment. There are two issues with which developers of case-based tutoring systems often contend: one is assessing and retrieving similar cases from the case library; the second one is delivering the case contents to the students. Whilst an earlier paper has addressed the former issue, this paper focuses on the latter by defining a computational mechanism that is used for delivering the case content. The mechanism is developed by defining a procedural semantics on the case graph which incorporates the dynamic modelling capability of petri nets. A case is initially opaque to the student. During case interaction, however, it will be made transparent gradually by engaging the students with problem-solving activities. The activities are modelled using the notions of marking places and firing transitions, where places and transitions represent case variables and operations, respectively. The idea is illustrated with an example of providing guidance to students solving problems in the domain of Newtonian mechanics.

Keywords: **Artificial Intelligence, Conceptual Graphs, Intelligent Tutoring Systems, Case-based Reasoning**

1 Introduction

This paper presents an extended theory of representing problem-solving cases proposed in [5] for the purpose of modelling instructional activities between the cases and the learners within the context of case-based tutoring systems (CBTS) [11]. In response to the classic criticisms [12] leveled at the first-generation of computer-assisted learning software that frequently have to go back to inflexible, pre-compiled problem solutions, CBTS is very attractive for several reasons. Two of them are particularly appealing to us. From an instructional perspective, students are highly influenced by past examples (i.e. real cases) to guide their problem-solving activities [1] or completing cognitive tasks [8]. Our project sponsor demands that the final system should faithfully reflect what students actually do when completing their homework. It is, therefore, our aim to ground our system design at the outset on sound psychological findings about pupils' learning behaviours. Secondly, from a technical viewpoint, case-based adaptation techniques are powerful in adapting interface components to the user's need [14].

Individual learner's needs, style and progress do differ substantially. Case-based reasoning technology [7] endows the system with the capability of inferring what is considered 'best' for the students by referring to their past learning histories. [5] proposed the use of conceptual graphs (CG) [13] for representing tutorial cases. While this method elegantly tackles the issue of assessing case similarity, how the case graphs are built remains a 'black-box'. The case users have no way to inspect the internal processes for constructing the graph. To ensure the cases are useful in tutorial contexts, the knowledge components of the cases need to be 'available' to the students. What we mean by 'available' is making the case solution transparent, i.e. the system is capable of justifying each problem-solving step being shown to the students in terms of the underlying physical principles.

The procedural semantics defined on case graphs which forms the core contents of this paper, provides a way of making the solution procedures explicit to the students. The idea is to synthesize a CG and the actor graph defined in [13] into one single global graph instead of treating them separately. The resulting structure is a tripartite graph that has three types of nodes: concept nodes, symbolic relation nodes and mathematical relation nodes. The mathematical relation nodes are for handling mathematical calculations in the domain of Newtonian mechanics, the targeted subject domain of our project These calculations are important in many science and engineering applications. In making the synthesis, two important ontological commitments were made. Firstly, human cognitive functions in studying a concrete case are viewed as a process of constructing graphs. Relevant concept nodes are created and linked to each other via some appropriate relation nodes (whether symbolic or mathematical). A case represented by the graph consists of sets of concept nodes and relation nodes, but to what extent the students understand the case contents remains unknown until some observable actions are seen. Secondly, the process of building the graph is based on the notion of concept node marking. Initially, the sets of nodes in a case are all opaque to the users because they are not yet marked. The set of nodes representing the initially given physical quantities are marked first. Each problem-solving step is viewed as generation of new graph nodes, but they are implemented as the nodes states change from unmarked to marked. To mark a set of nodes, the mathematical relation nodes (or operators) which link the marked and the unmarked nodes have to be fired. The procedures of solving the problem are defined as the firing sequence for marking the target concept nodes. The subgraph associated with a particular fired node represents the semantics of the knowledge behind its firing.

2 Formal Definition of the Case Constituents

We represent a typical case abstractly by a directed graph which is composed of

- ∗ Three disjoint sets of vertices C, R *and* R_m (i.e. $C \cap R = \emptyset$; $C \cap R_m = \emptyset$; $R \cap R_m = \emptyset$ and $C \cap R \cap R_m = \emptyset$) where C represents the set of concept nodes; R represents the set of symbolic relation nodes; and R_m represents the set of mathematical relation nodes.
- ∗ A set of directed arcs E such that $E \subseteq (C \times R) \cup (R \times C)$. Each arc $e \in E$ connects a concept $c \in C$ to a symbolic relation $r \in R$ or vice versa.
- ∗ A set of directed arcs E_m such that $E_m \subseteq (C \times R_m) \cup (R_m \times C)$. Each arc $e_m \in E_m$ connects a concept $c \in C$ to a mathematical relation $r_m \in R_m$ or vice versa.

Shown in Figure 1 is an example case graph where

$C = \{c_1, c_2, c_3, c_4, c_5, c_6\}$;
$R = \{r_1, r_2, r_3, r_4\}$;
$R_m = \{r_{m1}, r_{m2}, r_{m3}, r_{m4}\}$;
$E = \{(c_1, r_1), (r_1, c_2), (c_3, r_2), (c_4, r_2), (r_2, c_2), (c_5, r_4), (r_4, c_2), (c_6, r_3), (r_3, c_1)\}$; and
$E_m = \{(c_1, r_{m1}), (r_{m1}, c_2), (c_2, r_{m2}), (c_2, r_{m3}), (c_5, r_{m3}), (r_{m2}, c_3), (r_{m3}, c_4), (c_3, r_{m4}), (c_4, r_{m4}), (r_{m4}, c_6)\}$.

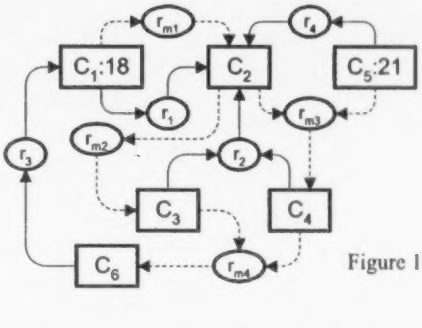


Figure 1

- ∗ For every $r_m \in R_m$, there exist an input set $I_r(r_m)$ and an output set $O_r(r_m)$ such that

$I_r(r_m) = \{ c \in C \mid (c, r_m) \in E_m\}$; (c , r_m) is called the input arc of r_m, and c is called the input concept of r_m; and
$O_r(r_m) = \{ c \in C \mid (r_m , c) \in E_m\}$; (r_m, c) is called the output arc of r_m, and c is called the output concept of r_m.

For example, the input/output set of the node r_{m3} in Figure 1 are $I_r(r_{m3}) = \{c_2, c_5\}$ and $O_r(r_{m3}) = \{c_4\}$ respectively.

- For every $c \in C$, it is defined as ***marked*** if it is being instantiated to a specific individual. In Figure 1, c_1 and c_5 are marked whereas the others are non-marked.
- The marking μ of a graph G can be represented by a *n-vector*: $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$, where each $\mu_i \in (T, F)$. For example, the graph in Figure 1 has the marking $\mu = (T, F, F, F, T, F)$.
- A mathematical relation node $r_m \in R_m$ is *enabled* whenever each concept $c \in I_r(r_m)$ is marked. In Figure 1, only r_{m1} is enabled at that marking.

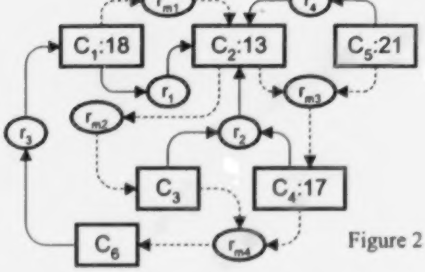


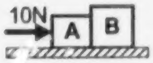
Figure 2

- When a mathematical relation node is enabled, it can be fired at any time and every time a mathematical relation is fired, every $c \in O_r(r_m)$ will be marked[1].
- For every $c \in O_r(r'_m)$, where r'_m is the fired mathematical relation, the content of c is evaluated according to the formulas inscripted in the respective $r'_m \in I_c(c)$.
- Supposing the formulas inscripted in r_{m1} is $c_1 = c_2 + 5$ and r_{m3} is $(c_2 + c_5) / 2$, the firing of r_{m1} will mark c_2 which enables r_{m3} because c_5 has already been marked. If r_{m3} is fired later, a new marking (shown in Figure 2) will be formed and become $\mu = (T, T, F, T, T, F)$.

3 Representing Mechanics Problem-solving Cases

In our application domain, Newtonian mechanics, two categories of physical entities are identified with respect to the cases we use for tutoring: *physical objects* and *physics concepts*. Both are represented, however, as rectangular-shaped concept nodes. In each case, a number of physical objects are involved, such as a block, a car, a plane, a spring, etc., but they are normally described abstractly just as a physical object. Various meaningful relations obtain between the objects, which essentially represent the physical configuration between them. For instance, it makes sense to represent the 'rest_on' relation that holds between a block and a plane whenever the block is on the plane. Other meaningful relationships are: 'above', 'contact_with', 'moves_on', and so forth. There are attributes, intrinsic and motion-related, of the physical objects which refer to one object only. For example, 'acceleration'(a motion-related attribute) and 'mass' (an intrinsic attribute) applies to a single physical object on its own. In representing a physical situation, there are some other domain-related ideas such as external force or friction, which characterize the case being described. All these concepts are categorized as physics concepts as they are used to describe the state of the world depicted by the case. Figure 3 shows a typical case adopted from a standard physics textbook.

Two blocks A & B are resting on a frictionless horizontal plane as shown. If an external force of 10N is acting on A, what is the acceleration of the blocks and the force of contact between them? (The masses of A and B are 3kg and 7kg respectively).



---

[1] The notion of marking and firing is borrowed from the petri nets formalism [9]

Solution: Apply Newton's 2nd Law on A&B

$$\begin{aligned}
\text{Net Force}_{A\&B} &= \text{Mass}_{A\&B} \times \text{Acceleration}_{A\&B} \\
\text{External Force}_{A\&B} &= \text{Mass}_{A\&B} \times \text{Acceleration}_{A\&B} \\
10 &= (3+7)\ \text{Acceleration}_{A\&B} \\
\text{Acceleration}_{A\&B} &= 1\ \text{m/s}^2
\end{aligned}$$

Apply Newton's 2nd Law on A

$$\begin{aligned}
\text{Net Force}_{A} &= \text{Mass}_{A} \times \text{Acceleration}_{A} \\
\text{External Force}_{A} + \text{Contact Force}_{A} &= \text{Mass}_{A} \times \text{Acceleration}_{A} \\
10 + \text{Contact Force}_{A} &= 3 \times 1 \\
\text{Contact Force}_{A} &= -7\text{N}
\end{aligned}$$

Apply Newton's 2nd Law on B

$$\begin{aligned}
\text{Net Force}_{B} &= \text{Mass}_{B} \times \text{Acceleration}_{B} \\
\text{Contact Force}_{B} &= \text{Mass}_{B} \times \text{Acceleration}_{B} \\
\text{Contact Force}_{B} &= 7 \times 1 \\
\text{Contact Force}_{B} &= 7\ \text{N}
\end{aligned}$$

Figure 3: A typical Newtonian mechanics case and its solution

As the complete graph representing the case occupies too much space, the whole graph is divided into several subgraphs. To illustrate the idea, three representative subgraphs are shown in Figure 4, 5 and 6. The subgraph in Figure 4 represents the physical objects involved in the case and their relationships. The (component) nodes encodes the part-whole relationship between the whole system A&B and its constituents A and B. The tuple [Blocks: A&B] → (component) → [Block: B] depicts the block labelled as 'B' as part of the whole system labelled as 'A&B'. The other relation nodes essentially represent the spatial relationships between the objects.

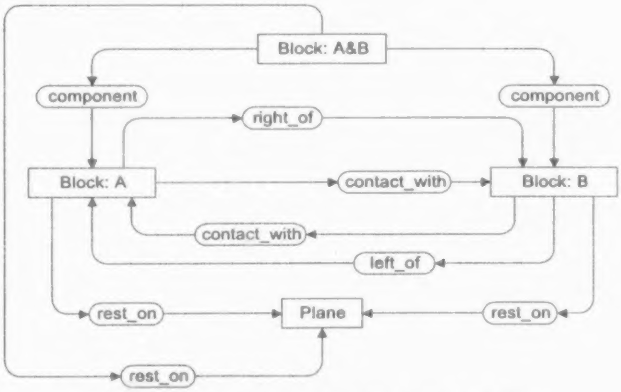


Figure 4: The subgraph showing the physical objects involved in the case and their relationship

The subgraph shown in Figure 5 concerns the attributes, both intrinsic and motion-relationed, of block A, and other relevant physical concepts centred around it. The absurd type [**T**] as the agent of the Net_Force_A and External_Force_A indicates it is something that is of no relevance to us. In Figure 6, those concept types that participate in some sort of mathematical relations are shown. Note that most of the arcs in Figure 6 are dotted indicating they are different from the usual symbolic relations.

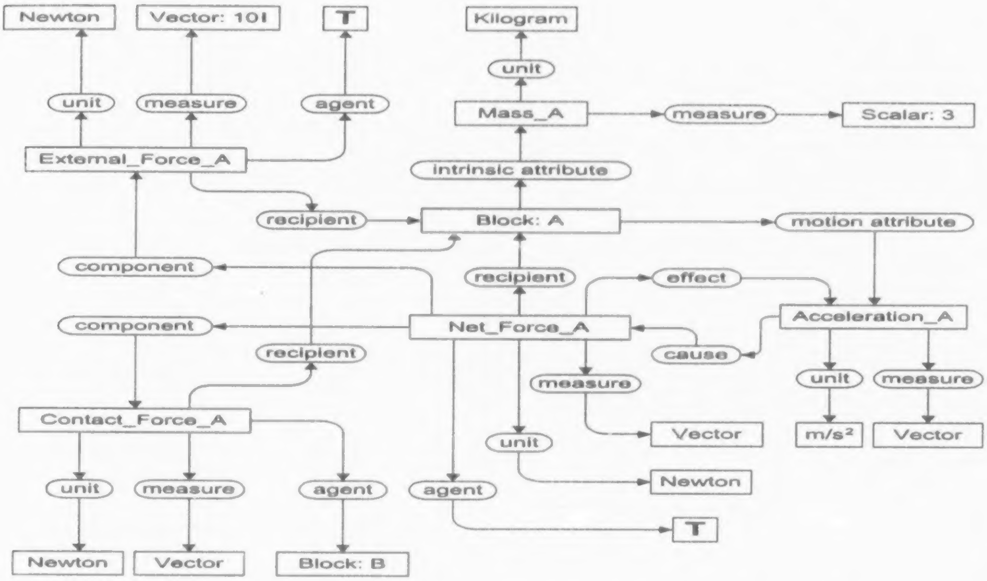

Newton
Vector: 10I
T
Kilogram
unit
unit
measure
agent
Mass_A
measure
Scalar: 3
External_Force_A
intrinsic attribute
recipient
Block: A
motion attribute
component
recipient
effect
component
Net_Force_A
Acceleration_A
recipient
cause
unit
measure
measure
Contact_Force_A
Vector
m/s^2
Vector
unit
Newton
unit
measure
agent
agent
T
Newton
Vector
Block: B

Figure 5: Subgraph showing the attributes of block A and other relevant physical concepts.

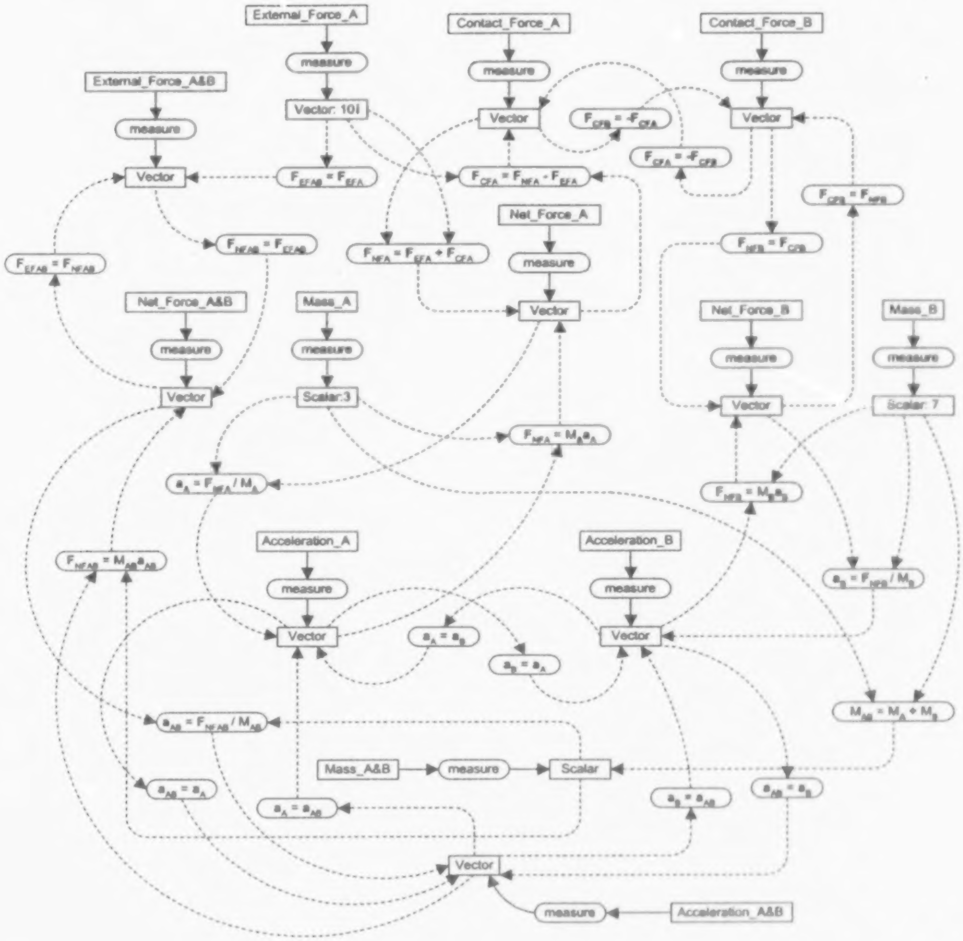

External_Force_A
Contact_Force_A
Contact_Force_B
External_Force_A&B
measure
Vector: 10i
Vector
$F_{CFB} = -F_{CFA}$
$F_{CFA} = -F_{CFB}$
$F_{EFAB} = F_{EFA}$
$F_{CFA} = F_{NFA} - F_{EFA}$
$F_{CFB} = F_{NFB}$
Net_Force_A
$F_{NFAB} = F_{EFAB}$
$F_{NFA} = F_{EFA} + F_{CFA}$
$F_{NFB} = F_{CFB}$
$F_{EFAB} = F_{NFAB}$
Net_Force_A&B
Mass_A
Net_Force_B
Mass_B
Scalar:3
Scalar: 7
$F_{NFA} = M_A a_A$
$a_A = F_{NFA} / M_A$
$F_{NFB} = M_B a_B$
Acceleration_A
Acceleration_B
$F_{NFAB} = M_{AB} a_{AB}$
$a_B = F_{NFB} / M_B$
$a_A = a_B$
$a_B = a_A$
$M_{AB} = M_A + M_B$
$a_{AB} = F_{NFAB} / M_{AB}$
Mass_A&B
Scalar
$a_{AB} = a_A$
$a_A = a_{AB}$
$a_B = a_{AB}$
$a_{AB} = a_B$
Acceleration_A&B

Figure 6: Subgraph showing the mathematical relationships between the relevant concept types

4 Modelling Variables Instantiation as Node Marking

Once a case has been encoded with the formalism, problem-solving activities can be modelled. When given a problem to tackle, the students will generally be asked for a new value from a set of given data. This is modelled as marking the concept nodes such as C_1 and C_5 in Figure 1. The goal is to get the concept node C_6 marked. At the initial marking, only r_{m1} is enabled and therefore any attempt to trigger other mathematical operations is not allowed and, thereby, invites tutorial intervention. The whole process of creating successive markings can be illustrated with a search tree (see Figure 7). The tree

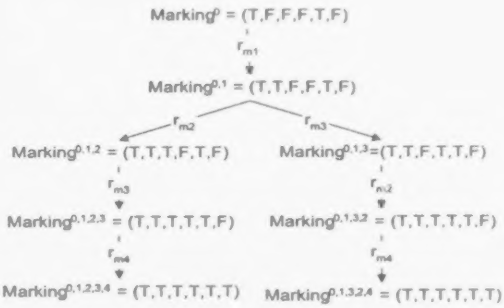

$Marking^{0} = (T,F,F,F,T,F)$
r_{m1}
$Marking^{0.1} = (T,T,F,F,T,F)$
r_{m2}
r_{m3}
$Marking^{0.1.2} = (T,T,T,F,T,F)$
$Marking^{0.1.3} = (T,T,F,T,T,F)$
r_{m3}
r_{m2}
$Marking^{0.1.2.3} = (T,T,T,T,T,F)$
$Marking^{0.1.3.2} = (T,T,T,T,T,F)$
r_{m4}
r_{m4}
$Marking^{0.1.2.3.4} = (T,T,T,T,T,T)$
$Marking^{0.1.3.2.4} = (T,T,T,T,T,T)$

Figure 7

indicates the student can gain access to a large solution space for him/her to explore but in the mean time the tutor can keep track of what can/cannot be done.

5 CLASP: A Case-based Learning Assistant System in Physics

A system called CLASP, has been developed to implement the idea. At the current stage of development, two types of activities associated with examples have been identified: providing solutions for studying, and exercises with answers; hence the modes of interaction in the CLASP prototype are also designed around these two themes. When the users issue a request (in terms of the problem description of their own problems) the system will search through its whole case library and provide them cases which match their request. The style of presenting the case will follow the user's wishes, but only two modes of interaction (solution studying and guided-problem-solving) are available. This is to reflect the common way of using examples in physics textbooks. In the study mode, the system presents the whole case (i.e. both the problem and solution statements) for the user to study. This looks like an electronic reference book and the student may browse through the relevant cases. In the guided-problem-solving mode, the system only presents the problem situation to the users, but appropriate system guidance will be provided in solving the problems. The schematic architecture of CLASP is shown in Figure 8. The students interact with the system with the support of the back-end knowledge base.

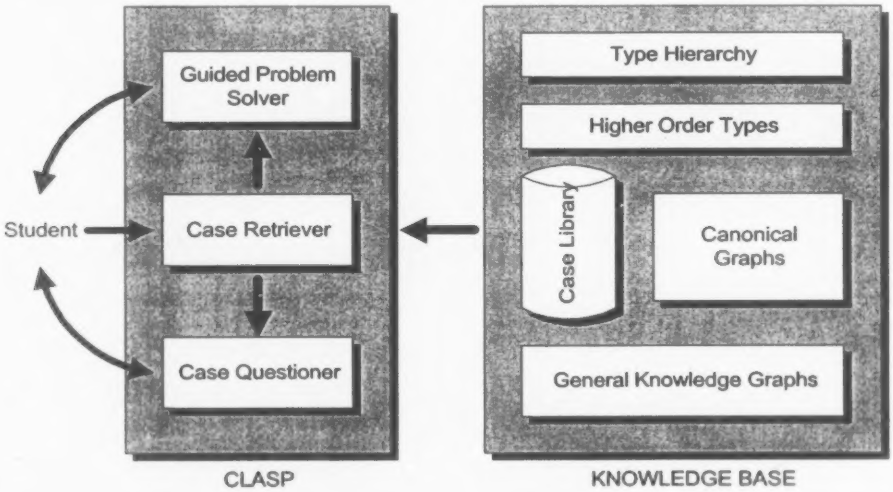


Figure 8: Schematic description of the CLASP architecture

Problem-solving in CLASP is modelled as a graph search. When a problem situation, such as the one shown in Figure 3, is encountered, the initial data are represented as concept nodes being instantiated to specific values and they are displayed to students on the working pad (Figure 9). Now the problem-solver can start tackling the problem by searching through the graph and seeing what additional information can be inferred from the initial given data. For the system to perform the tasks, the expertise has already been encoded in the case graphs, therefore the next step to be taken is searching the graph to find out which operators can be fired. The inferred steps may be unfolded or kept hidden for a while as a hint to advise the student. The intelligence of the system's problem-solving ability comes from its inference engine, being implemented by different graph search methods.

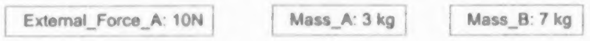
Problem Space	System's Comments:
	Initial Conditions: M_A = 3kg; M_B = 7kg; and External Force $_A$ = 10N.

Figure 9: The working pad and the corresponding system responds

The explanatory capability of the system comes from the matching of the input-operator-output nodes with the consequences of the general knowledge graphs. Whenever an operator is fired, the associated nodes will be matched against the consequences of the general knowledge graphs. If one is found, and it should be, then that particular graph will be tagged. If the student requests a justification of the step taken, the system can explain the graph in general terms. For example, the firing of an algebraic summation operator on the values of masses of two physical objects will match the consequence of the general knowledge graph in Figure 8 so the whole graph can be retrieved for explanation (Figure 10). The working pad, showing the problem space, and the explanation combinations supply the integration of what and why the step happened and the whole process becomes transparent to the student.

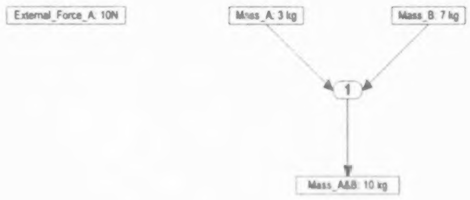
Problem Space	System's Comments:
 External_Force_A: 10N Mass_A: 3 kg Mass_B: 7 kg 1 Mass_A&B: 10 kg	Step 1: For a system comprising two components, the mass of the whole system is evaluated by the algebraic sum of the masses of their individual components. $M_{A\&B} = M_A + M_B$

Figure 10: The working pad and the corresponding system responds

6 Conclusions

Case-based reasoning (CBR) is a versatile AI technology and can been found in many industrial applications [2] but its potential in training and education is still not fully explored. The work reported here may serves to strengthen the position of CBR in developing instructional systems.

The contribution of the paper to the endeavour of computer-assisted learning is twofold. Firstly, technically, a formal framework for representing cases for learning purposes has been developed. Its formal basis provides a solid foundation for developing robust computer-based instructional systems. With this methodology, the developers only have to concentrate their effort on collecting and encoding the cases. The rest (generating relevant instructional activities from the cases) will be taken care of by the system. This approach offers another advantage for rendering the cases amenable to further analysis. This may be used for providing tool to verify the case-base for internal consistency. Secondly, educationally, our approach paves the way for systematic educational software engineering because it is built on the needs of users, not the technical skills of the developers. Often, educational software developers have adopted a technically-driven design philosophy. Such systems run the risk of losing sight of what is actually happening in the real learning setting.

Our approach avoids the temptation of jumping onto the hi-tech bandwagon but, instead, concentrates firstly on what the students really need. The reason we developed a case-based learning system was not due to the existence of the technology and trying to find what role the technology can play in learning. Rather, we choose to develop a case-based approach to learning because students do learn from referring to past cases. This principle we consider crucial in determining if the final system proves itself useful to our students. Other features of the system have not been described due to space limitation. They include generating different categories of questions from a case graph [6] to promote self-explanation from the students. The model proposed in this paper can also perform qualitative reasoning [4], and causal order between system variables can be represented succinctly.

References

[1] Anderson, J.R. et al., "Learning to Program in LISP", Cognitive Science, Vol. 8, pp.87-129, (1984).

[2] Bergmann, R. et al., "Developing Industrial Case-based Reasoning Applications", Springer-Verlag, (1999).

[3] Fung, P.W., "Designing an Intelligent Case-based Learning Environment with Conceptual Graphs", PhD Thesis, Dept. of Computation, UMIST, England, (1996).

[4] Fung, P.W., "Generating Qualitative Predictive Problems in a Case-based Physics Tutor" In B. du Boulary & R. Mizoguchi (Eds.) Artificial Intelligence in Education: Frontiers in Artificial Intelligence and Applications Vol. 39, Ohmsha: IOS Press, (1997).

[5] Fung, P.W. & Adam, A., "Conceptual Graphs for Similarity Measurement in a Case-based Physics Problem Solver" In Proceedings of AI-ED 95 – World Conference on Artificial Intelligence in Education. Washington D.C, (1995).

[6] Fung, P.W. & Adam, A., "Questioning Students on the Contents of Example Solutions" In Proceedings of European Conference on Artificial Intelligence in Education, Lisbon, (1996).

[7] Kolodner, J. "Case-Based Reasoning" San Mateo, CA: Morgan Kaufmann Publishers, (1993).

[8] LeFevre, J.A. & Dixon, P., "Do Written Instruction need Examples?" Cognition and Instruction, Vol.3, pp.1-30, (1986).

[9] Peterson, J.L., "Petri Net Theory and the Modeling of Systems." Englewood Cliffs. NJ: Prentice-Hall Inc. (1981).

[10] Sanders, K.E., Kettler, B.P. & Hendler, J.A. "The Case for Graph-Structured Representations" In D.B. Leake & E. Plaza (Eds.) Case-based Reasoning Research and Development. Lecture Notes in Artificial Intelligence, Vol. 1266, pp. 245-54, Springer-Verlag, (1997).

[11] Schank, R.C., "Case-based Teaching: Four experiences in Educational Software Design." Report No.7, The Institute of Learning Sciences, Northwestern University, (1991).

[12] Self, J., "Student Models in Computer-aided Instruction." International Journal of Man-Machine Studies, Vol. 6, pp. 261-76, (1974)

[13] Sowa, J., "Conceptual Structures: Information Processing in Mind and Machine" Reading, MA: Addison-Wesley, (1984).

[14] Specht, M. & Weber, G., "Episodic Adaptation in Learning Environments" In Proceedings of the European Conference on Artificial Intelligence in Education, Lisbon, pp.171-176, (1996).

Use of abstraction levels in the design of intelligent tutoring systems

Ruddy Lelouche
Département d'informatique
UNIVERSITÉ LAVAL Québec G1K 7P4 CANADA
Tel.: (+1-418) 656 2131, ext. 2597 Fax: (+1-418) 656 2324
e-mail: `lelouche@ift.ulaval.ca`

In problem-solving domains (mathematics, physics, engineering, and most "exact science" disciplines), the knowledge to be acquired by the student is twofold: the knowledge describing the domain itself, but also and mainly the knowledge necessary to solve problems in that domain. As a result, an educational system in such a domain encompasses three knowledge types: the domain knowledge and the problem-solving knowledge, i.e. the knowledge to be acquired and mastered by the student, and the tutoring knowledge, used by the system to facilitate the student's learning process. In this paper, we show how these three knowledge types can be modelled, how they should interact with one another in order to fulfil the system educational purpose, and above all how abstraction levels can shed a uniformizing light on the system operation and make it more user-friendly. We thus hope to bring some contribution to the general and important problem of finding a generic architecture to intelligent educational systems.

Keywords: **Intelligent tutoring systems, Abstraction, Complexity, System Design.**

1 Introduction

Teaching is a very complex process in itself. Teaching strategies and activities vary considerably: by the role and autonomy they give to the learner, by the type of interactions they trigger with him/her, by the evaluations they enable, by the relationships they make between theory and practice, etc. From that last perspective, teachable domains can be classified according to the type of knowledge to be acquired by the student: "know", "know-how", and "know-how-to-be". Examples of such domain types are respectively: anatomy or a language grammar, the skill to solve a mathematical or medical problem, and the capability to adapt to one's environment or to deal with personal relationships. We are more particularly interested in the second type.

Moreover, almost all teachable domains vary in complexity, from simple basics to relatively complex problems to solve or situations to deal with. Thus, a student should learn and master the basics of such a domain before being taught wider notions. And when a human tutor detects errors or misunderstandings, s/he usually draws the student's attention on a small subset of the involved knowledge, so that s/he may correct his/her errors and/or misunderstandings, focusing either on a given set of the domain knowledge or on the scope of knowledge involved by a given problem.

Problem-solving (PS) domains are the ones in which we are interested here. In such a domain, the knowledge to be acquired by the student is twofold: the domain knowledge itself, but also and mainly the knowledge necessary to solve problems in that domain. As a result, an education-oriented system in such a domain, which we here call a PS-ITS, must encompass three knowledge types: the domain knowledge and the problem-solving knowledge, constituting the knowledge to be acquired and mastered by the student, and the tutoring knowledge, used by the system to facilitate the student's learning process.

This paper has two goals: to present each of the three types of knowledge involved in a PS-ITS, and for each type of knowledge, to show how abstraction and complexity levels appear and how we think it is possible to deal with them.

To do so, we present in section 2 our domain knowledge modelling and how we exemplify it in a few PS domains . Next, in section 3, we focus on the advantage of separating the problem-solving knowledge from the domain knowledge in a PS-ITS, and we present some problem-solving activities in various domains. In section 4, we briefly describe some principles of tutoring knowledge modelling in a PS-ITS. In each of these three sections, we show how to use abstraction and complexity levels, exemplifying them in a few typical domains.

Finally, section 5 presents the educational interests of using abstraction and complexity levels when modelling the three types of knowledge involved in a PS-ITS.

2 Domain knowledge

In order to describe the domain knowledge, we first present its characteristics in a general PS-ITS (section 2.1). We then show how we model it in a few PS domains (section 2.2), and how such an approach lets us introduce the notions of abstraction and complexity levels (section 2.3).

2.1 General

The first type of knowledge involved in every ITS, the *domain knowledge* (DK), contains all theoretical and factual aspects of the knowledge to be taught to the student. Although its specific structure can be varied, it typically may include concepts, entities, and relations about the domain [Brodie & al., 1984], object classes and instances [Kim & Lochovsky, 1989], possible use restrictions, facts, rules, [Kowalski, 1979; Clocksin & Mellish, 1981], semantic or associative networks [Findler, 1979; Sowa, 1984], etc.

The main system activities centred on this knowledge type are:

- providing the student with theoretical presentations and explanations about the various knowledge elements and their relationships in the teaching domain;
- providing the other modules of the ITS, i.e. problem-solving and tutoring, with the necessary background of domain knowledge that they need.

2.2 Application to a few domains

In the particular domain of **cost engineering**, Lelouche and Morin [1997; Morin, 1998] represent this type of knowledge with concepts, relations, and a special case of relations modelled as concepts, the factors.

Concepts can be basic entities like investment, interest, investment duration, present and future values, compounding, compounding period, interest rate, annuity, etc.

Concepts are linked to one another by various types of *relations*: either usual knowledge-representation relations, like *subclass of*, *element of*, *sort of*, etc., or numerical relations represented by formulæ. Such a formula is:

$$F = P \times (1 + i)^n \tag{1}$$

which, given the present value P of an investment over n periods at interest rate i, computes the corresponding future value F of that investment.

A formula such as (1) can be rewritten as:

$$F = P \times \Phi_{PF,i,n} \text{ where } \Phi_{PF,i,n} = (1 + i)^n \tag{2}$$

$$P = F \times \Phi_{FP,i,n} \text{ where } \Phi_{FP,i,n} = (1 + i)^{-n} \tag{3}$$

thus introducing the *factors* $\Phi_{PF,i,n}$ and $\Phi_{FP,i,n}$. Factors allow us to separate their definition (rightmost equalities above, a quantitative aspect) from their possible uses in the application domain (leftmost equalities, a qualitative aspect).

Similarly, the factor $\Phi_{AP,i,n}$ converts a series of identical annual amounts A into a unique present value P:

$$P = A \times \Phi_{AP,i,n} \text{ where } \Phi_{AP,i,n} = \frac{(1+i)^n - 1}{i\,(1+i)^n} \tag{4}$$

Actually, $\Phi_{AP,i,n}$ is a sum of Φ_{FP} factors (see details below). The factor $\Phi_{PA,i,n}$ does the reverse process:

$$A = P \times \Phi_{PA,i,n} \text{ where } \Phi_{PA,i,n} = \frac{i\,(1+i)^n}{(1+i)^n - 1} \tag{5}$$

There exist other factors converting gradient and geometrical series of amounts into a present or future value; such factors are also computed as a sum of $\Phi_{FP,i,n}$ factors.

In **geometry**, *concepts* are basic elements like point, line, segment, and later more elaborate elements like angle, then square, rectangle, circle, ellipse, polygon, solid, polyhedra, etc. Examples of *relations* between concepts are adjacency (of segments or angles), parallelism (of lines or line segments), complementarity (of angles), etc. Upper-level, more abstract concepts are then defined using lower-level ones, as well as relations between these lower-level concepts (e.g. a triangle is a set of three segments adjacent pairwise).

In **mechanical physics**, we similarly introduce *concepts* like time, distance, velocity, acceleration, mass, force, and later angle, angular velocity, angular acceleration, moment of inertia, torque, etc. We also introduce *relations* like the one defining velocity as the variation in distance per unit of time, or the one stating that the acceleration a is proportional to the force F that is applied. Introducing a generalization from linear to rotational movement, another relation defines angular velocity as the angle variation per time unit, and another one states that the angular acceleration α of a solid body is proportional to the torque τ that is applied to it. More precisely, we have:

for a linear movement $F = M \times a$ where M = total mass of the body (6)

and for a rotational movement $\tau = I \times \alpha$ where $I = \sum (m \times r^2)$ (7)

Equation (6) expresses Newton's second law. In equation (7), I is the moment of inertia and is expressed in terms of the mass m of each of its particles and of its distance r to the rotation axis. Obviously M in equation (6) and I in equation (7) play the role of factors as in cost engineering.

Although formulæ like (2-7) related to factors essentially involve quantitative aspects, the similarities and differences between them, and the circumstances regulating the use of either one, are of a deeply qualitative ground. In **cost engineering**, if the *value* of factors is indeed calculated from two or three numerical parameters, the *context* in which they are defined depends on whether we have to timewise move a unique amount or a series of amounts, identical or not, or conversely to compute an equivalent annual amount, etc. In fact, this context corresponds to the type of conditions that govern the investment, or *investment conditions type*, without respect to the amounts and durations involved, and is thus essentially qualitative. Similarly, in **physics**, the proportionality between force and linear acceleration, or between torque and angular acceleration, expresses a qualitative relationship. Only if the need arises, the exact relationship can be expressed by using the actual mass M in formula (6) or the result of the computation of the moment of inertia I in formula (7), which in the general case involves a simple or double integral. Indeed, did not the use of qualitative reasoning originate with qualitative physics?

2.3 Towards the notions of abstraction and complexity levels

In most PS-domains, abstraction most obviously appears in the *definition of the domain concepts* themselves, like we showed in all three domains above.

If *factors* are used in the domain, it also appears that every factor introduces an additional intermediate abstraction level between the concepts implied in the equation defining it. For example, in the case of formula (1), or equivalently formulæ (2) and (3) in **cost engineering**, or in the case of formula (6) and (7) in **physics**, we have (see figure 1):

- at the bottom of the hierarchy, basic concepts "making technicalities explicit" if necessary: the interest rate and the number of periods in cost engineering, the distribution of mass within the body volume in physics ;
- above them, concepts more fundamentally related to the problem being solved, namely in cost engineering the present and future values of the investment, and in physics the force and acceleration, or the torque and angular acceleration;
- between these two levels, an intermediate level created by the introduction of the factor (Φ_{FP}, Φ_{PF}, M, or I).

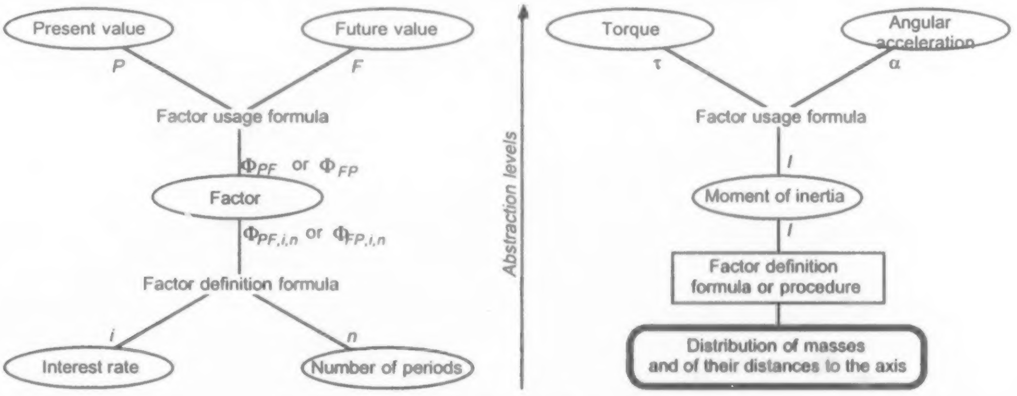


Figure 1 — Representation of a factor as a concept.

That intermediate status of the factor, originally just an intermediate variable in computations [see formulæ (2) and (3) or (6) and (7)], makes it appear as a pedagogically oriented concept, which clearly separates

- the computational, quantitative aspect of the factor definition,
- from the practical, qualitative aspect of the factor usage in a domain problem.

This follows the theory [Lenat & al., 1979; Malec, 1989] according to which the use of multiple *abstraction levels* eases the modelling process and simplifies inferences which may be made on the domain concepts.

Most interestingly, our scaffolding approach can be made more general, at least in certain domains, where we may present and use higher-level factors built upon these first ones. Indeed, in cost engineering, "above" Φ_{FP} and

Φ_{PF}, the factors used to express the present and future values of a series of identical amounts (and vice versa) are a first way to generalize this concept hierarchy. For example, the Φ_{AP} factor is indeed a sum of Φ_{FP} factors:

$$\Phi_{AP,i,n} = \sum_{k=1}^{n}\Phi_{FP,i,k} = \sum_{k=1}^{n}(1+i)^{-k} = \frac{(1+i)^n - 1}{i\,(1+i)^n}$$

where the last expression results from computing the geometrical series shown. This example constitutes a proof of (4), but also and mainly shows that the Φ_{AP} factor is at a higher level than Φ_{FP}. Note that this refers to a *complexity level* rather than an abstraction level, since it is due to the way the Φ_{AP} factor is defined and computed. Similarly, the moment of inertia of a complex body can be (and often is) computed as the sum of elementary moments of inertia, and therefore is at a higher complexity level.

3 Problem-solving knowledge

In order to describe the problem-solving knowledge, we now present the general characteristics regarding problem-solving knowledge modelling in a PS-ITS (section 3.1). As we did in section 2, we then examplify our model in the cost-engineering and physics domains (section 3.2).

3.1 General

The second type of knowledge is specific to PS-domains [Ganeshan & al., 2000; Gertner & VanLehn, 2000], henceforth to PS-ITSs. We call it *problem-solving knowledge* (PSK). It contains all inferential processes used to solve a problem resulting from the instantiation of a practical situation based on the domain knowledge [Kowalski, 1979; Patel & Kinshuk, 1997]. In other words, in order to be able to solve a problem, the problem-solving knowledge needs a theoretical background, which is found in the domain knowledge. The processes stored in PSK may be represented in various ways, using any or all of: logic [Kowalski, 1979], procedural networks [Brown & Burton, 1978], semantic networks with procedural attachments, (augmented) transition networks, production rules [Goldstein, 1979; Anderson & Reiser, 1985], etc.

The main system activities centred on this knowledge type are:

- providing the inferential tools for problem solving, by the system or by a student;
- providing the inferential tools for coaching a student in a problem-solving session.

The main advantage of separating the problem-solving knowledge from the domain knowledge is that it emphasizes the distinction between the domain itself and the skills used to solve a practical problem in that domain, thus simplifying the learning process. That knowledge separation into DK and PSK is common to all PS-domains; this is why we believe that PS-ITSs, which are aimed at helping the student to learn how to solve problems, should display the same knowledge separation.

Besides, following [Lelouche & Morin, 1997], we can use — we believe in a novel way — that separation between DK and PSK to define four *generic operating modes* in a PS-ITS, based on the type of knowledge involved (DK or PSK), and on who "generates" that knowledge (the system or the student).

- In *domain-presentation mode*, the student asks the system some information about a domain theoretical element, and the system reacts by transferring to the student the required information or knowledge. The knowledge involved in this category is always DK, system-generated.
- In *demonstration mode*, the student asks the system to solve a practical problem or to coach him/her while s/he solves a problem. In the first case, the problem typically comes from the student him/herself, whereas in the latter one the problem typically comes from the tutoring system. In either case, the main level of knowledge involved is PSK, system-generated.
- In *domain-assessment mode*, the system prompts the student to develop a domain element, and the student thus expresses his/her understanding of that element. If judged necessary, the system may then intervene to correct that understanding. The knowledge involved in this mode is essentially DK, student-generated.
- Finally, in *exercising mode*, the system prompts the student to solve a practical problem. The student then solves it step by step, showing what s/he understands of the involved problem-solving knowledge and of the associated domain knowledge. If necessary, the system may decide to intervene in order to help him/her reach his/her goal or to correct it. The knowledge involved in this mode is naturally PSK, student-generated.

3.2 Application to a few domains

Several problem-solving activities are domain-independent, like:

1. identify and instantiate the given problem data;
2. identify and instantiate the expected result(s);
3. apply a formula;
4. apply a theorem.

Every PS-domain also has its own domain-dependent activities. For example, in **cost engineering**, we have:

5. draw a temporal diagram to represent the relevant events;
6. compare amounts located at the same date;
7. compare amounts located at different dates;
8. add amounts situated at the same date;
9. add amounts situated at different dates;
10. choose a reference date;
11. move an amount from one date to another;
12. collapse a series of periodic amounts into one single amount;
13. explode an amount into a series of periodic amounts.

Similarly, in the subset of **mechanical physics** refered to above, some activities would be:

14. compute a torque;
15. compute an angular acceleration;
16. compute a moment of inertia.

In many cases, a PS activity can be rephrased into, restated as, a different one, of a lower *abstraction level*, because more immediate, more down-to-earth, closer to the problem to be solved. For example, in **mechanical physics**, assuming that the torque and the moment of inertia of a given solid body are known (either given or previously computed), the activity "compute the angular acceleration" (activity 15) would be expressed as, or translated into "apply formula (7)", an instance of the lower-level activity 3. A PS physics tutor is presented in [Gertner & VanLehn, 2000].

Sometimes, a PS task may also be divided into smaller ones, letting us use again the notion of *complexity levels* in these tasks. For example, in **cost engineering**, comparing two amounts situated at different dates implies:

- first, choosing a reference date at which to make the comparison;
- then, moving either (or both) amount(s) from its (their) present date(s) to the reference date;
- finally, comparing the amounts, now both located at the same reference date.

These subactivities (of types 10, 11, and 6 respectively in the sample list above) thus appear to be of a lower complexity level than the initial one (of type 7). However, it is interesting to note that, although activity 7 turns out to be more complex than activity 6 (the latter is part of the former), both are stated using the same *abstraction level*.

It may also happen that some lower-level activities can *only* appear as components of a higher-level one. For example, still in **cost engineering**, the activity "drawing a temporal diagram" (type 5 above) implies the following tasks, which can only be accomplished as part of that activity (hence their identification in this paper from 5a to 5d):

5a. draw a timeline to encompass all periods implied by the problem data;
5b. draw arrows representing the amounts involved in the problem data;
5c. if necessary, split an amount (or each amount in a series) to simplify the computations;
5d. qualitatively draw a special arrow to represent the expected result of the computations to be made.

In that case, activity 5 is both of a higher complexity level and of a higher abstraction level than any of its subactivities.

4 Tutoring knowledge

We now briefly present the *tutoring knowledge* (TK) in order to help the reader to better apprehend the relationships of that knowledge with DK and PSK. This third type of knowledge contains all tutoring processes enclosed in the ITS. It is not directly related to the teaching domain or to problem solving, but is there to help the student understand, assimilate, and master the knowledge included in DK and PSK [Gagné & al., 1992; Gagné & Trudel, 1996].

The main system activities using TK are:

- ordering and formatting the topics to be presented to the student;
- monitoring a tutoring session, i.e., triggering the various tutoring processes according to the system tutoring goal and the student's actions; such monitoring may imply giving explanations, asking questions, changing to another type of interaction, etc.;
- in a PS-domain, while the student is solving an exercise, monitoring the student's PS activities: understanding and assessing these activities, giving advice to correct or optimize them, giving hints or partly solving the exercise at hand (as required by the student or by the tutoring module), etc.;
- continuously analysing the student's progress in order to improve the efficiency of the tutoring process.

The advantage of separating the tutoring knowledge from the knowledge of the domain to be taught has been emphasized long ago [Goldstein, 1977; Sleeman & Brown, 1982; Clancey, 1986; Wenger, 1987], and lies in the reusability of TK in various domains. In the case of PS-domains, the domain to be taught clearly encompasses both DK and PSK; indeed, the term "domain knowledge" applies to DK if referring to the knowledge type, and to DK + PSK if referring to the knowledge to be acquired. Therefore, as shown in the introduction, in a PS-ITS, knowledge ends up being separated into three categories rather than two.

We believe that the tutoring processes are triggered by *tutoring goals* which depend on the current educational setting and learning context. The role of tutoring goals has been discussed in several works, some of the most recent ones dealing with task and instruction ontologies [Mizoguchi, 1999]. In the current state of our research, our assumption is the following: the underlying hierarchy or hierarchies governing the way tutoring processes interact with one another is not related to these processes *per se*, but rather to the *current goal* to be attained when they are invoked. The current goal varies during the session, depending on the student's actions or difficulties, following a dynamically built abstraction-based hierarchy. If our assumption turns out to hold, then the *dynamic structure of educational goals and subgoals* — which itself depends on the student's desires or abilities, the main underlying objective of the tutoring system, the student's state (e.g. of tiredness, etc.) and performance, etc. — will determine the succession of tutoring processes activated and tutoring interactions taking place. To our knowledge, the use of abstraction levels to induce a dynamic hierarchy of tutoring goals is new, as is the assumption that such a hierarchy will play a major role in activating the various tutoring processes and student–system interactions. Learning goals have been used by Towle [2000], but only for educational simulations, not for tutoring processes in general.

5 Educational interests of abstraction and complexity levels

In the above sections, we have sketched a complexity– and abstraction–level approach to help model the three types of knowledge involved in a PS-ITS. In this section, after clarifying these notions in section 5.1, we present the educational interests of our model. Sections 5.2 to 5.4 focus on the type of knowledge respectively presented in sections 2 to 4. Section 5.5 summarizes that discussion with some overall pedagogical interests of our approach.

5.1 An informal definition of abstraction and complexity levels

In the first three sections, we only refered to abstraction and complexity levels. Here, we try to define these notions better and in a more generally applicable way. Both notions are based on the common notion of *refinement*, but differ in how the refinement is made: in a general way, abstraction is based on, or refers to, *expressiveness* or *scope*, whereas complexity is based on, or refers to, the *number of components*.

For *concepts*, taking geometry as an example, a polygon has a higher abstraction level than a triangle or a square, because the number of sides in a polygon is indefinite, but a lower abstraction level than a set of segments, because these segments in a polygon are forced to be pairwise adjacent; a square has a higher complexity level than a triangle, because it has more sides, and also because there are constraints (re. size and angles) between these sides. In cost engineering, we saw that the factors Φ_{FP} and Φ_{AP} are expressed at the same abstraction level, although Φ_{AP} has a higher complexity level, because of the way it is defined and computed. A similar distinction between abstraction levels and complexity levels holds for the *relations* they express.

For *problem-solving activities*, we have similar distinctions, as shown in section 3.2 with several examples.

Finally, the same holds for *tutoring processes*, or student–system interactions. For instance, the ITS task of tutoring a student while s/he is solving a problem will turn out to be of a higher complexity level if the student encounters more difficulties, although the abstraction level of this process does not depend on the particular student being tutored or on the particular problem being solved. On the other hand, reacting to a student request for hint, or for explanation, is of a lower abstraction level than the previous one; however, there again, the complexity of that task will depend on the specific student request (some simply formulated questions may have quite complex answers!), and will eventually depend also on the way the student is or is not satisfied with the initial system response.

Such level-based distinctions have also been made, for example, by Mizoguchi [1999]. Note that, although the statement "A has a higher abstraction level than B" is clear and may be true, we think that the number of abstraction levels between A and B is not defined, because that number would depend on the modelling effected; for the same reason, it would be even more meaningless to try to assign a numeric value to these levels.

5.2 Domain modelling

The definition of concepts from the simplest to the most complex induces a long-time known *presentation order* for the subject matter. Similarly and in addition, the factor hierarchy described in section 2 for cost engineering lets us derive an *order for the presentation of factors* to the student, from the lowest (simplest) level up to the highest, i.e. with increasing understanding complexity. That does not imply that such an order is unique, or even the best (e.g. a student's personal interests might make another order more motivating for him/her), but it is justified by our model. This presentation order may itself induce, like for domain concepts, a possible *order for prerequisites*; e.g., if a student experiments difficulties to deal with Φ_{AP}, has s/he well mastered Φ_{FP}, a conceptually simpler factor?

Moreover, the factor-induced intermediate abstraction levels will permit the ITS to exhibit a *sharper modelling of conceptual errors*. For example, the source of an understanding error concerning one of the two relations in equation (2) or (3) or (7) (see also figure 1) is much easier to identify using the corresponding factor, either as a definition error or as a usage error, than an error concerning the global equation (1), where the definition and application relationships are not made explicit, and therefore are impossible to distinguish. Similarly, an error using a Φ_{AP} factor may be diagnosed as possibly resulting from an insufficient mastery of the simpler factor Φ_{FP} as concept (which in turn will be diagnosed as related either to its definition, or to its usage). Similarly in physics, if the student stumbles on concepts like angular acceleration or moment of inertia, has s/he mastered the simpler although similar concepts of acceleration or mass?

Abstraction and complexity levels on domain elements (concepts and relations, possibly including factors) can then be used to introduce various abstraction levels of explanations. Such explanations can then be tailored to the student's questions, and adapted to the reminders possibly needed by the student.

5.3 Problem-solving modelling

The problem-solving activities briefly presented in section 3 naturally display abstraction and complexity levels. Indeed, a standard problem can usually be divided, possibly in more than one way, into major steps, which can then be split into simpler substeps. As explained in 5.1, each subactivity in that case may be either simpler (lower complexity level) or more concrete (lower abstraction level) than the original one, or both.

In a first development stage, these abstraction and complexity hierarchies, both for domain elements and for problems to be solved, can ease the *definition of exercise types* to be implemented into the ITS, and can ease the tutor module task of *choosing the exercise type* to challenge the student with. Later, once that basic system is operational, the same hierarchies can help develop an automatic *exercise generator* dealing with the domain elements to be mastered by the student. That approach will then help the student to acquire a better critical mind about the relative importance of *problem–solving* knowledge vs. *domain* knowledge.

As for domain elements, abstraction and complexity levels can be used to introduce various types of *explanations* about the problem to be solved, varying both in abstraction (focus level, terms used, references made) and in complexity (quantity of details, possible references to the problem substeps). Moreover, our approach will lead the student to focus specifically on the *activities* for which s/he needs more tutoring, with the abstraction and complexity levels appropriate to his/her individual case.

5.4 Tutoring modelling

Functioning mode		Domain-presentation mode	Demonstration mode	Domain-assessment mode	Exercising mode
Main type of knowledge involved		Domain knowledge	Problem-solving knowledge	Domain knowledge	Problem-solving knowledge
Student's main goal		To learn (acquire or improve knowledge)		To assess his/her learning	
Direction of the knowledge transfer		System → Student		Student → System	
Typical interaction	Trigger (start)	The student asks the system...		The system prompts the student...	
		some information about a domain theoretical element	to solve a practical problem or to coach him/her in problem solving	to develop a domain element	to solve a practical problem
	Knowledge exchange	The system presents...		The student presents his/her view of...	
		the requested element	a possible solution to the requested problem	the requested element	a possible solution to the given problem
	Result (closure)	The student expresses his/her understanding...		The system assesses the student's answers, and possibly corrects them	
		of the element	of the problem solution		

Table 2 — Characteristics of the four typical operating modes of a problem-solving ITS.

As presented in section 3.1, the distinction between DK and PSK leads to the natural definition of four operating modes. Their main characteristics are recalled in Table 2.

The successive tutoring goals aimed at by the system (see section 4) are likely to result in a chain of recursive calls of the tutoring processes invoked. This recursivity will or will not be direct, depending on the tutoring interaction types being chained: the system might decide to temporarily change between interaction types, e.g. to respond to the student's actions or requests. However, the potential length of this chain is only apparent: because of the abstraction hierarchy of tutoring goals, each newly invoked process will be called with a *narrower scope* and/or a *lower complexity*, which eliminates the risk of "forgetting" the initial tutoring goal or of running into an infinite loop.

More generally, tutoring the student may take the form of explanations, guidance, hinting, or even partially solving the exercise on which the student is currently working. The level at which these will be conducted will depend on the current tutoring goal (see section 4). We think our approach is close to that of VanLehn and his colleagues [2000], although they focused their attention on fading and deepening (a particular result of the tutoring interactions) rather than on the current pedagogical goal (the cause for these interactions).

5.5 Overall interests of these abstraction and complexity levels

Abstraction levels are certainly not new. What we think is new is to use them in a systematic way to shed a uniformizing light on the ITS design and operation, and to make it more user-friendly once implemented.

First, they may help to give a better tailoring to the system tutorial interventions to fulfil the student's needs and the system tutoring goals, thus improving its conviviality and efficiency.

Then, all the capabilities presented above should result in smoother, more "natural", human-like interactions with the student. This improved ability to reproduce a human teacher's behaviour contributes again to make the system more user-friendly, and more likely to be used by the student.

Finally, although that aspect is not in the scope of this paper, our refinement of the three types of knowledge as described in sections 2 to 4 paves the way to the conception and the implementation of a structured error model, and eventually of a structured student model.

6 Conclusion

This presentation of a possible knowledge structure for PS-domains, which emphasizes the separation between domain knowledge and problem-solving knowledge, shows how a general functioning theory of such an ITS — namely the four operating modes described in sections 3.1 and 5.4 — can naturally be derived.

Moreover, the abstraction and complexity levels highlighted throughout this paper can be used as a common guideline to help finding an appropriate representation for each one of the three knowledge type, and thus can help creating more efficient ITSs. More generally, this guideline can shed a uniformizing light on the system design, although it has never been used in a systematic way in the design or implementation of an ITS.

We thus hope to bring some contribution to the general and important problem of finding a generic architecture for intelligent tutoring systems.

References

[1] Anderson John R.: *Cognitive Modelling and Intelligent Tutoring*. Washington, DC: National Science Foundation (1986).

[2] Anderson John R. & B. J. Reiser: "The LISP Tutor". *Byte* **10**, no. 4, p. 159-175 (1985).

[3] Brodie Michael L., John Mylopoulos & Joachim W. Schmidt, eds.: *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*. New York: Springer Verlag (1984).

[4] Brown John S. & Richard R. Burton: "Diagnostic models for procedural bugs in mathematical skills". *Cognitive Science* **2**, p. 155-192 (1978).

[5] Clancey W. J.: "Qualitative student models". *Annual Review of Computer Science* **1**, p. 381-450 (1986).

[6] Clocksin William F. & Christopher S. Mellish: *Programming in Prolog*. Berlin: Springer-Verlag (1981).

[7] Findler Nicholas V., ed.: *Associative Networks*. Orlando, FL: Academic Press (1979).

[8] Frasson Claude, Gilles Gauthier & Alan Lesgold, eds.: *Intelligent Tutoring Systems*, Proc. of the *3rd International Conference, ITS'96*, Montréal (Canada), 12-14 June 1996. LNCS 1086, Berlin: Springer (1996).

[9] Gagné Denis & André Trudel: "A highly flexible student-driven architecture for computer-based instruction". [Frasson & al., 1996], p. 66-74 (1996).

[10] Gagné Robert M., Leslie J. Briggs & Walter W. Wager: *Principles of Instructional Design*, fourth edition (1st edition published in 1974). Orlando, FL: Harcourt Brace Jovanovich (1992).

[11] Ganeshan R., W.L. Johnson, E. Shaw & B.P. Wood: "Tutoring diagnostic problem solving". In [Gauthier & al., 2000], p. 33-42 (2000).

[12] Gertner A.S. & K. VanLehn: "ANDES: a coached problem-solving environment for physics". In [Gauthier & al., 2000], p. 133-142 (2000).

[13] Goldstein Ira P.: *The Computer as Coach: an Athletic Paradigm for Intellectual Education*. AI Memo 389, AI Laboratory. Boston, MA: Massachusetts Institute of Technology (1977).

[14] Goldstein Ira P.: "The genetic epistemology of rule systems". *International Journal of Man-Machine Studies*, Vol. 1, No. 1, p. 51-77 (1979).

[15] Gauthier Gilles, Claude Frasson & Kurt VanLehn, eds.: *Intelligent Tutoring Systems*, Proc. of the *5th International Conference*, Montréal (Canada), 19-23 June 2000. LNCS 1839, Berlin: Springer (2000).

[16] Kim Won & Frederick H. Lochovsky, eds.: *Object-Oriented Concepts, Databases, and Applications*. ACM Press, Reading, MA: Addison-Wesley Publ. (1989).

[17] Kowalski R.: *Logic for Problem Solving*. Berlin: North-Holland (1979).

[18] Lelouche Ruddy & Jean-François Morin: "Introduction of pedagogical concepts in domain modelling for an ITS". *European Journal of Engineering Education*, Vol. 23, No. 2, p. 255-271 (June 1998).

[19] Lelouche R. & J.-F. Morin: "Knowledge types and tutoring interactions in an ITS in a problem-solving domain". *FLAIRS 1997, Proc. of the 10th International FLorida Artificial Intelligence Research Symposium*, Daytona Beach (Florida, U.S.A.), 10-14 May 1997, p. 62-66 (1997).

[20] Lenat D., F. Hayes-Roth & P. Klahr: *Cognitive Economy*. Working Paper HPP-79-15, Stanford Heuristic Programming Project, 46 pages. Stanford, CA: Stanford University (June 1979).

[21] Malec J.: " Knowledge elicitation during dynamic scene description ". *ACM–SIGART Newsletter: special issue on knowledge acquisition*, No. 108, p. 162-163 (April 1989).

[22] Mizoguchi Riichiro: "Ontology-awareness for intelligent instructional systems". Invited talk. *Advanced Research in Computers and Communications in Education, Volume 1* (G. Cumming, T. Okamoto & L. Gomez, eds.), Proceedings of the *7th International Conference on Computers in Education*, *ICCE 99*, Chiba (Japan), 4-7 November 1999. Amsterdam, Netherlands: IOS Press, p. 45-52 (1999).

[23] Morin J.-F.: *Conception of an intelligent tutoring system in cost engineering: knowledge representation, pedagogical interactions, and system operation*. Master Thesis, Département d'Informatique. Québec, Canada: Université Laval (1998).

[24] Patel Ashok & Kinshuk: "Intelligent tutoring tools in a computer-integrated learning environment for introductory numeric disciplines". *Innovations in Education and Training International*, Vol. 34, No. 3, p. 200-207 (1997).

[25] Sleeman D. H. & John Seely Brown, eds. : *Intelligent Tutoring Systems*. London: Academic Press (1982).

[26] Sowa John F.: *Conceptual Structures*. Reading, MA: Addison-Wesley (1984).

[27] Towle B.: "Using student task and learning goals to drive the construction of an authoring tool for educational simulations". In [Gauthier & al., 2000], p. 173-181 (2000).

[28] VanLehn K., R. Freedman, P. Jordan, C. Murray, R. Osan, M. Ringenberg, C. Rosé, K. Schulze, R. Shelby, D. Treacy, A. Weinstein & M. Wintersgill: "Fading and deepening: the next steps for ANDES and other model-tracing tutors". In [Gauthier & al., 2000], p. 474-483 (2000).

[29] Wenger, Étienne: *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann (1987).

Using Decision Networks for Adaptive Tutoring

Peng-Kiat Pek* and Kim-Leng Poh**
** Singapore Polytechnic*
500 Dover Road, Singapore 139651
E-mail: baiychen@magix.com.sg
*** National University of Singapore*
10 Kent Ridge Crescent, Singapore 119260
E-mail: isepohkl@nus.edu.sg

This paper reports a research project that uses dynamic decision networks in providing teacher with information on students' misconceptions and students with online tutoring. A set of Bayesian networks models the conditional dependencies between learning objectives and goals which are associated with the curriculum. Student's responses to test items are recorded and transformed as evidence into a relevant Bayesian network to compute his likely state of knowledge mastery. The personalized Bayesian network is then converted into a dynamic decision network by adding utility and decision nodes. Tutoring policy is followed through and necessary responses from the student are solicited using additional test items. The student Bayesian network is updated when new evidence arrives, and is again converted to a decision network to determine the next tutoring policy. This process is repeated until the pre-requisites are achieved. The results generated by the system and future directions are discussed.

Keywords: **Adaptive Tutoring, Decision Network, Student Model, Tutoring Strategy**

1 Introduction

Tutoring of students is an ill-structured problem that is characterized by:
(a) Uncertainty of student's knowledge mastery.
(b) Preferences, judgements, intuition, and experience of teacher.
(c) Criteria for decisions are occasionally in conflict, and highly dependent on the teacher's perception.
(d) Decisions must be achieved in limited time.
(e) The student's mental states evolve rapidly.

This study attempts to address these issues by using an intelligent *decision-theoretic* approach. The framework of this research has contributed to the development of an intelligent decision support system called *iTutor*, for tutoring Engineering Mechanics at Singapore Polytechnic.

Probabilistic or Bayesian networks [9] and decision analysis [5] have shown to be capable of solving many real-world problems involving reasoning and decision marketing under uncertainty. Bayes's nets allow for efficient reasoning and inference about combination of uncertain evidence. Student modeling with Bayes's nets for intelligent tutoring had achieved successes, see for example in [16], [11], and [2]. The differences in these works lie mainly in the choice of variables and granularity of the models.

In Villano's Knowledge Space Theory, the basic unit of knowledge is an *item* (in the form of a question). The student's knowledge state is defined as the collection of items that the student is capable of answering. The collection of all feasible states is called the *knowledge structure*, and it is connected by the *learning path*. By incorporating uncertainty at each node, the knowledge space can be transformed into a Bayes's net. The Bayes's net then constitutes a student model where probabilistic reasoning can be performed when evidence is available. Reye on the other hand, uses pre-requisite relationship of domain knowledge and *dynamic belief network* for modeling student's mastery of a topic. Finally, Conati and Vanlehn make use of teacher's solution(s) as the ideal model to track student' s faulty knowledge as the student solves a problem.

Our work here differs from others in that we construct relevant Bayes's nets by modeling learning objectives (L), evidence (V) from student responses, application of knowledge to different situations (C), and learning goal (G). A decision network [3] is then formed by adding decision and utility nodes to the Bayes's net. As it is computationally intractable to track student' s solution in real time, we use sequential decisions to generate tutoring strategy that anticipates students' responses.

This paper is organized as follows: Section 2 provides an overview of the conceptual framework for the decision theoretic intelligent tutoring system called *iTutor*. The transformation of student's responses to evidence is discussed in Section 3. Section 4 illustrates how the student model is constructed from a set of Bayes's nets, while Section 5 presents the tutoring strategy model using two-step look-ahead decision network. The results of a typical *iTutor* session are illustrated in Section 6. It emphasizes the automation of decision network construction and shows that when student's responses are available, the system is able to diagnose student's misconceptions and to provide adaptive tutoring using the generated strategy. Finally, we conclude by discussing future directions.

2 Framework of Adaptive Tutoring

Figure 1 shows the essential components of adaptive tutoring in *iTutor*.

The ***Evidence Model*** converts the student response (x_{ijk}) to item *i* into evidence of knowledge mastery for a relevant learning objective (v_{jk}).

The ***Student Model*** consists of a set of Bayes's nets with nodes that are either Evidence, Case, Learning Objective, or Goal. These nodes are initialized with prior information from the teacher' s judgement and theoretical probability models. The student model can be subsequently updated to reflect a student' s knowledge mastery when evidence is available.

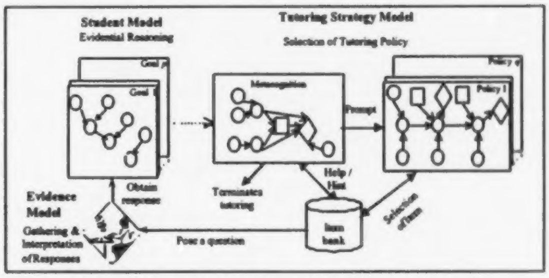


Figure 1: Inferencing Kernel of Adaptive Tutoring

The ***Tutoring Strategy Model*** uses decision-theoretic approach to select *satisfying* [14] learning objectives for tutoring student. The metacognition sub-module determines the appropriate tutor' s action: providing more help or hint, prompting another question, or stop the tutoring session. Dynamic Decision Network (DDN) provides approximate solutions for partially observable Markov decision problems, where the degree of approximation depends on the amount of look-ahead. If the decision is to obtain evidence of mastery on a learning objective, an item of difficulty b_i that matches the student' s ability θ will be selected. Student's response is collected, evaluated, and transformed into evidence at the relevant nodes in the student model. The *chance* nodes in DDN are updated and a decision policy is generated. In this way, the system is able to adapt tutoring to the needs of the student and achieve the objectives of the curriculum.

3 Evidence Model

The student' s responses are processed in the evidence model. Let V_{jk} be the evidence node that indicates the student' s (j) mastery state of learning objective k. Let X be the set of responses and $x_{ijk} \in X_{ik} \subseteq X$ be the response to item i which tests the k^{th} learning objective, then

$$\Pr(v_{jk} \mid x_{ijk}) \propto \Pr(v_{jk}) \prod_i \Pr(x_{ijk} \mid v_{jk})$$

where $\Pr(v_{jk})$ is the prior probability which can be obtained statistically from past data. $\Pr(x_{ijk} \mid v_{jk})$ is the likelihood of correct-answer score. An example of the likelihood function is $\delta_{ik} \exp(-b_i v_{jk})$ where δ_{ik} is the importance of knowing learning objective k so as to answer item i correctly and b_i is the difficulty index for item i.

4 The Student Model

The Student Model consists of a set of Bayes's nets, and each Bayes's net models the student' s mastery of a key concept (*goal*). In Section 4.1, the structure of the student model is defined. The construction of Bayes's net and the conditional probability assignment are discussed in Section 4.2. Instantiation of an evidence node activates a message passing process in the Bayes's net. This process results in the updating of marginal probabilities at the nodes. Most commercial software for developing probabilistic network possesses efficient algorithm [1] for implementing the message passing process.

4.1 Semantics of the Student Model

The Student Model is a directed acyclic graph (DAG) that represents a joint probability distribution of a key concept and several learning objectives. A node represents the learning objective as a random variable, and an arc represents possible probabilistic relevance or dependency between the variables. When there is no arc linking two nodes, it indicates probabilistic independence between the variables. In this study, the variables are classified into four types Evidence, Case, Learning Objective, and Goal as shown in Figure 2.

More formally, a student model in *iTutor* is a DAG $\Im = (N, \psi)$ where $N = N_v \cup N_L \cup N_C \cup N_G$ are the nodes such that N_v is a set of evidence nodes, N_L is a set of learning objective nodes, N_C is a set of case nodes, and N_G is a set of goal nodes.

Symbol	Name	Function
N_v	Evidence Node	It contains knowledge states based on student' s response.
N_C	Case Node	It contains knowledge states that reflect ability to apply knowledge in different situations (cases).
N_L	Learning Objective Node	It contains knowledge states of key learning objectives (defined in the syllabus).
N_G	Goal Node	The concept student is expected to know. Each Bayes' s net must have at least one Goal node.

Figure 2: Types of Nodes in Student Model

$\psi = \psi_{pL} \cup \psi_{pC} \cup \psi_{pG}$ are the arcs such that $\psi_{pL} \subseteq N \times N_L$ are arcs into learning objective nodes, $\psi_{pC} \subseteq N_v \times N_C$ are arcs from evidence nodes to case nodes, and $\psi_{pG} \subseteq (N_L \cup N_G) \times N_G$ are arcs from learning objective or goal nodes to the goal nodes.

Notice that evidence nodes have no parent node and only evidence nodes could be the parents of case nodes. Goal nodes are always sink nodes and they have parents that are either learning objective nodes or goal nodes. This signifies that mastery of a concept (goal node) is dependent on the mastery of learning objective(s) and/or pre-requisites (other goal nodes).

4.2 Construction of a Bayes's Net

Figure 3 shows a Bayes's net on mastery of a hypothetical concept (*goal*) "XYZ". Each node has three knowledge states: *non-mastery, partial-mastery,* and *mastery*. The granularity of Bayes's net depends on the number of nodes and its states. However, as the granularity becomes finer, the number of entries in the conditional probability table grows exponentially.

Values at the root nodes are known as *prior* probabilities while that at other nodes are *conditional* probabilities. To use the probabilistic network the random variables must be initialized with prior probability values. These values may be based on teacher's belief or past statistics. An intuitive method is to generate a probability table based on seven-category of the difficulty of learning objectives (see Table 1). These probability values are to be input as the prior probability of the related evidence. The teacher also has the flexibility to amend the values based on their belief and context of usage. On the other hand, the probability values can be obtained from statistics of previous tests/examinations. A simple procedure for the use of past statistics is:

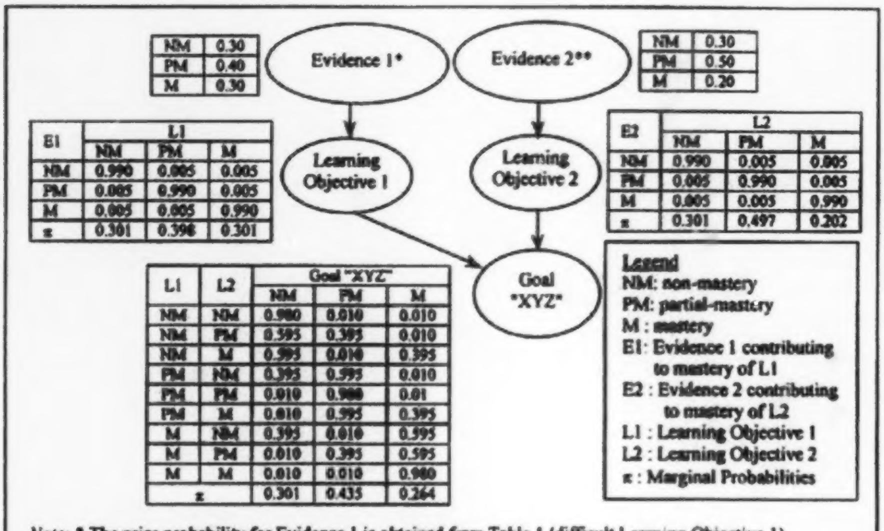
a) Assigned learning objectives to each question;

Table 1 Category of Difficulty in Mastering the Learning Objective

	Probability values		
Category	**NM**	**PM**	**M**
very easy	0.001	0.009	0.99
easy	0.01	0.09	0.90
fairly easy	0.05	0.15	0.80
Neutral	0.10	0.20	0.70
fairly difficult	0.20	0.30	0.50
difficult	0.30	0.40	0.30
very difficult	0.40	0.50	0.10

b) Enter student's responses (in terms of percentage) for the questions that she has answered;

c) Compute the average number of students (in percentage) for each mastery category: 040 (non-masterystate), 40-70 (partial-mastery state), and 70-100 (mastery state).



NM	0.30
PM	0.40
M	0.30

NM	0.30
PM	0.50
M	0.20

E1	L1		
	NM	PM	M
NM	0.990	0.005	0.005
PM	0.005	0.990	0.005
M	0.005	0.005	0.990
π	0.301	0.398	0.301

E2	L2		
	NM	PM	M
NM	0.990	0.005	0.005
PM	0.005	0.990	0.005
M	0.005	0.005	0.990
π	0.301	0.497	0.202

L1	L2	Goal "XYZ"		
		NM	PM	M
NM	NM	0.980	0.010	0.010
NM	PM	0.595	0.395	0.010
NM	M	0.595	0.010	0.395
PM	NM	0.395	0.595	0.010
PM	PM	0.010	0.980	0.01
PM	M	0.010	0.595	0.395
M	NM	0.395	0.010	0.595
M	PM	0.010	0.395	0.595
M	M	0.010	0.010	0.980
π		0.301	0.435	0.264

Legend
NM: non-mastery
PM: partial-mastery
M : mastery
E1: Evidence 1 contributing to mastery of L1
E2 : Evidence 2 contributing to mastery of L2
L1 : Learning Objective 1
L2 : Learning Objective 2
π : Marginal Probabilities

Note: * The prior probability for Evidence 1 is obtained from Table 1 (difficult Learning Objective 1)
** The prior probability for Evidence 2 is obtained from statistics

If a probability distribution function is able to describe the statistics, it can be used. In Figure 3, the values Pr(E2=non-mastery) = 0.30, Pr(E2=partial-mastery) = 0.50, and Pr(E2=mastery) = 0.20 are obtained from statistical data for this particular evidence. It is acceptable for another person to assign different probability values so long as it is consistent with the probability axioms [12]. Since the decision theory approach is normative rather than descriptive, it is able to explain the actions of the decision-maker.

For any node n_q, the conditional probability required to specify the Bayes's net is computed based on the relative importance (*weights*) of the parent nodes $pa(n_q)$ to itself.
If the state of n_q and pa(n_q) is the same, then $\Pr(n_q \mid \mathrm{pa}(n_q)) = \sum_{\mathrm{pa}(n_q)} (w_{pq} - (c-1)\kappa)$

$$\text{else } \Pr(n_q \mid \mathrm{pa}(n_q)) = \sum_{\mathrm{pa}(n_q)} \kappa \qquad (1)$$

where *c is the number of states and* $0 \le w_{pq} \le 1$.

κ is a constant and a measure of uncertainty such as careless errors, lucky guesses, changes in the student knowledge state due to learning and forgetting, and patterns of student responses unanticipated by the designer of the student model. The weights w_{pq} are either assessed based on the teacher's subjective judgment or past students' responses to closely related items.

Referring to Figure 3, since Learning_Objective_1 is dependent only on Evidence_1, $w_l = 1$. Let Learning_Objective_1 has greater influence on mastery of goal "XYZ" than Learning_Objective_2, $w_{1g} = 0.6$, and $w_{2g} = 0.4$. Assigning $\kappa = 0.005$, the conditional probability tables can be computed using equation (1).

5 Tutoring Strategy

When a student logon to *iTutor*, the system automatically searches his ability index from the database. The

BEST COPY AVAILABLE

ability index is either computed from the tests taken previously by the students, or from her knowledge states in the student model (see Section 5.1). Human tutors consider the student' s emotional state in deciding how to respond. Similarly in *iTutor*, the system considers factors such as response time, response pattern, student knowledge structure to determine tutoring actions: give more hint, help, ask another question, or stop the tutoring session. If the decision is to prompt another item, a learning objective and an appropriate item will be selected to coach her (see Section 5.2). Section 5.3 discussed the generation of tutoring strategy based on student' s response.

5.1 Mapping of Knowledge State to Student Ability

Let the student' s ability be $\theta_j = (\theta_{j1}, \theta_{j2}, ..\theta_{jm}, ..\theta_{jp})$. A function $f: v_{jm} \rightarrow \theta_m$ where v_{jm} is the evidence at the goal node (g) of m^{th} Bayes' s net. An example of such function is:

$$\theta_{jm} = \begin{cases} N\,(1.5, 0.6) & v_{jm} \geq 0.7 \\ N\,(0.5, 1) & 0.4 < v_{jm} < 0.7 \\ N\,(-1, 1.2) & v_{jm} \leq 0.4 \end{cases}$$

where N (μ, σ) denotes a *normal* distribution with mean μ and standard deviation σ

The computed ability index is then used to categorize (Advance, Intermediate, or Beginner) the student. An appropriate learning objective is selected based on the heuristic shown in Table 2. Value assignment is used to compute the path length of Bayes's net and is used as preference for tutoring policy generation. They are as follow:

$$\begin{aligned} &\text{value}(G) = 0 \ \text{ for } G \in \{\text{Goal nodes}\} \text{and ch}(G) = \phi \\ &\text{value}(\text{ch}(N)) = 0 \text{ if ch}(N) = \phi \\ &\text{value}(N) = \text{value }(\text{ch}(N)) + 1 \ \text{ for node } N \\ &\text{where ch}(N) \text{ is the child node of } N \qquad (2) \end{aligned}$$

Table 2 Search Heuristic for Identifying Learning Objective for Coaching

Category	Identification of first Learning Objective for tutoring
Advance	25% of pathLength*
Intermediate	50% of pathLength
Beginner	75% of pathLength
New#	50% of pathLength ($\lceil$pathLength/2$\rceil$)

Note:

- * pathLength denotes max(value(N)) $\forall N \in$ Bayes's net.
  The value(N) assignment is stated in equation 2
- # The category *New* refers to students who login to the system the first time

5.2 Item Selection

Each item is tagged with an index (b_i) that estimates the minimum ability to answer it correctly with 0.5 probability. The items are assumed to be independent and the index obtained through statistic of past students' attempts or assigned using teacher' s belief. Subsequent update of item difficulty index may be performed through item response theory [4] such as Rasch model [10].

From the set of items related to a learning objective, an item i is selected based on: $\theta_j - b_i < \varepsilon$ where ε is a

Table 3 Utility of Various Outcomes

Condition / Expression	Preference
Decision: Stop	
$S(N)$ = "M" & value(N) = 0	1
$S(N)$ = "N" & value(N) = 0	0
k = number of $N_k \in \{(N_k, S_k)\}$ with same S_k	$1 - k/5$
number of tries, n, for the same learning objective	$1 - n/5$
Decision: Ask item on same N	
$S(N)$ = "M"	0
$S(N)$ = "N"	1
$S(N)$ = "P"	$\left(1 - \frac{\text{value(N)}}{\text{pathLength}}\right) \times \eta$ where η is a constant
Decision: Ask item on ch(N)	
$S(N)$ = "M"	1
$S(N)$ = "N"	0
$\gamma = \max(\Pr(S(\text{ch}(N)) = \text{"M"} \mid x = 1) - \Pr(S(\text{ch}(N)) = \text{"N"} \mid x = 0)$	
Decision: Ask item on pa(N)	
$S(N)$ = "M"	0
$S(N)$ = "N"	1
$\gamma = \max(\Pr(S(\text{pa}(N)) = \text{"M"} \mid x = 1) - \Pr(S(\text{pa}(N)) = \text{"}N\text{"} \mid x = 0))$	

Remarks: $S(N)$ denotes the knowledge state of node N
ch(N) denotes child node of node N

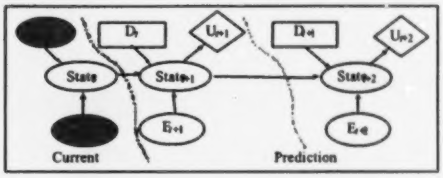


Figure 4: Dynamic Decision Network

Table 4 Utility Values for Item Difficulty Level Selection

Current Knowledge State (State$_t$)	Current Response (E$_t$)	Next Question Typ (D$_{t+1}$)		
		Easy	Ave.	Diff
Non-Mastery	Correct	-0.2	1	0
	Wrong	0.4	-0.2	-0.4
Partial Mastery	Correct	-0.4	0.2	1
	Wrong	0.4	1	-0.2
Mastery	Correct	-0.4	-0.2	-0.2
	Wrong	-0.2	0.2	-0.2

pre-defined small value. This ensures selected item is challenging and likely to be solved by the student. Teacher's solution will be displayed upon student's request so that she can learn from her mistake. This strategy assumes student's ability is dynamic and can be raised to higher levels through self-paced computer-aided tutoring.

5.3 Tutoring Policy Generation

To bring the probabilistic network one step closer to being a useful *intelligent tutoring system*, automated decision-making capability has been added. When asked to provide a tutoring policy for the student, the system generates a course of action based on her current mastery states. The tutoring policy aims to use a series of items with differing difficulty to determine more precisely her mastery of specific learning objectives. Items are categorized into *easy*, *average* and *difficult*. In this project, a two-step look-ahead dynamic decision network is recommended so as to compromise between the need to invoke policy generation routine for a decision and the long computing time to generate policy with many decisions.

Define a Bayes's net with nodes N,

E_t be the evidence at time t,

$\mathrm{Bel}(N)$ be a vector of probabilities (updated over time),

d_t be the decision at time t, and

α be the normalization constant.

Prediction 1: $\mathrm{Bel}^*(N_{t+1}) \leftarrow \sum_{N_t} \Pr(N_{t+1} \mid N_t, d_t)\mathrm{Bel}(N_t)$

Estimation 1: $\mathrm{Bel}(N_{t+1}) \leftarrow \alpha \Pr(E_{t+1} \mid N_{t+1})\mathrm{Bel}^*(N_{t+1})$

Prediction 2:
$\mathrm{Bel}^*(N_{t+2}) \leftarrow \sum_{N_{t+1}} \Pr(N_{t+2} \mid N_{t+1}, d_{t+1})\mathrm{Bel}(N_{t+1})$

Estimation 2: $\mathrm{Bel}(N_{t+2}) \leftarrow \alpha \Pr(E_{t+2} \mid N_{t+2})\mathrm{Bel}^*(N_{t+2})$

Expected Utility : $\sum_{N_t} \mathrm{Bel}(N_t) \max_{d_t} \sum_{N_{t+1}} \Pr(N_{t+1} \mid N_t, d_t) \times$
$\max_{d_{t+1}} \sum_{N_{t+2}} \Pr(N_{t+2} \mid N_t, d_t, N_{t+1}, d_{t+1}) \times$
$U(N_t, d_t, N_{t+1}, d_{t+1}, N_{t+2})$

Figure 5: Prediction-estimation Process

Figure 4 shows a dynamic decision network (DDN) used in this study. In addition to the decision nodes for current and fiture time steps, the DDN also contains the previous decision, D_{t-1}, as an evidence node. When the evidence for state t arrives, the probability distributions of $State_t$ are updated [1] using the prediction-estimation process (see Figure 5). After the initial prediction of probabilities (Bel*), $State_{t+1}$ estimates the new belief based on projected evidence [13]. This process repeats for $State_{t+2}$. Eventually, the expected utility is evaluated by a sequence of summations and maximizations. Tables 3 and 4 show the utility functions for node U_{t+2}. Selecting the outcomes with maximum expected utility value constitute the tutoring policy.

6 An Illustration

6.1 Construction of a Decision Network

In this project, the construction of all probabilistic networks is performed using Netica API [7]. A module leader enters the learning objectives and the weights of the key concept *Forces* using Microsoft Access [6]. The probabilistic values shown in Figure 6 are entered based on past examination results. By clicking the button "Model Construction", a Bayes's net (see Figure 7) and a decision network (see Figure 8) on "Forces"

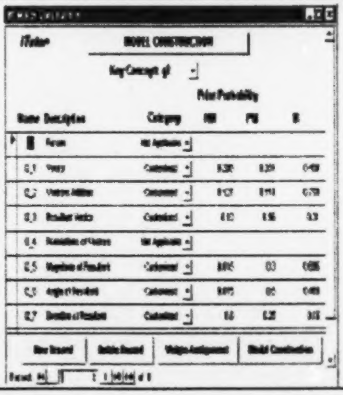


Figure 6: A Snapshot on Data Entry for Model Construction

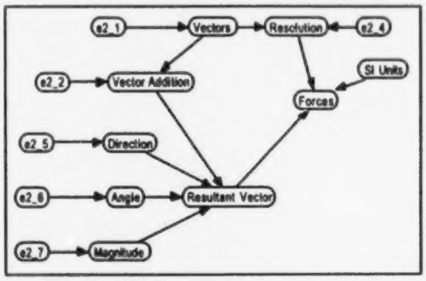


Figure 7: Bayes's Net on *Forces*

will be created. Teachers who are familiar with Netica application [8] can use the generated Bayes's net to perform *what-if* analysis. For example, a teacher may want to determine the likely student's improvement if he provides remedial instructions on "Resolutions of Vectors". He can do so by instantiating the evidence node *e2_4* to "Mastery" state, and observe the probability of mastery in the goal node labeled *Forces*.

6.2 Diagnosis of a Student's Misconceptions

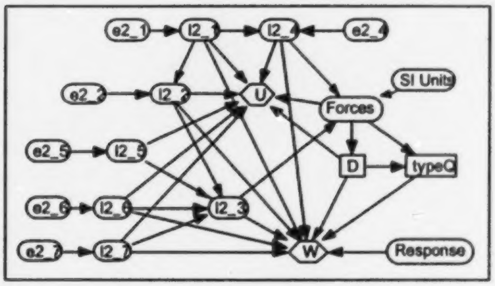


Figure 8: Decision Network on *Forces*

The items to be presented to the students are coded by the teacher using Scientific Notebook [15]. With *iTutor*, the teacher is able to monitor student's progress through the database management tool. Figure 9a shows a snapshot of a student who had answered item "Force_001" correctly and partially correct for item "Force_004". The teacher can track a student's mastery states by clicking the "Advice" button. The system transforms the responses to evidence, and instantiates the evidence nodes in the Bayes's net as shown in Figure 9b. The posterior mastery states are displayed (see Figure 9c). The output also provides the teacher information on specific learning objectives to tutor. In addition, he can also examine the detailed strategy by clicking the "Tutorial Strategy" button. This action causes the generation of a decision network (see Figure 9d). Figure 9e shows items to be posed to the student if she continues with the online tutorial. At any stage, the teacher may intervene by providing personal coaching.

7 Conclusions

Presently, the students' knowledge states remain unchanged until additional evidence is available. The system also uses a constant learning rate for all students. One future direction is to include additional parameters to model student forgetting and learning rates. Another area is to provide a user interface for teachers not familiar with Netica application to perform *what-if* analysis. In this way, the teacher will be able to focus on student's issues rather than to learn another software tool. The next future direction is to include probability functions other than *Normal* distribution. This is essential when the ability distribution of student cohort is not symmetric.

A significant result of this project is the use of Bayesian networks to generate sound probabilistic inferences. Another contribution is the automation of decision networks construction. The recommended strategy is used in adaptive tutoring. With *iTutor*, teacher is able to monitor the student's progress and yet had time for lesson preparation and coaching of weaker students. In addition, the teacher has accessed to the student's knowledge states and actions taken by *iTutor* at every stage of the tutoring process. Moreover, it enables students to have tutorials customized to their needs.

References

[1] Castillo, E., Gutiérrez, J. M., & Hadi, A.S., "Expert Systems and probabilistic network Models", Springer-Verlag, (1997).

[2] Conati C. & Vanlehn K., "Teaching meta-cognitive skills: Implementation and evaluation of a tutoring system to guide self-explanation while learning from examples", Proceedings of AIED '99, 9th World Conference of Artificial Intelligence and Education, Le Man, France, (1999).

[3] Deam, T. L. & Wellman, M. P., "Planning and Control", Morgan Kaufmann, San Mateo, California, (1991).

[4] Hambleton, R.K., Swaminathan, H., & Rogers, H. J., "Fundamentals of Item Response Theory", Sage Publications, Inc, (1991).

[5] Howard, R.A. & Matheson, J.E. "Readings on the Principles and Applications of Decision Analysis", Menlo Park, California: Strategic Decisions Group, (1983).

[6] Litwin, P., Getz, K., & Gilbert, M. , "Access 97 Developer's Handbook", 3rd edition, Sybex Inc, (1997).

[7] Norsys, "Netica API Programmer's Library Reference Manual", version 1.06, Norsys Software Corp., (1999)

[8] Norsys, "Netica Application Reference Manual", version 1.12, Norsys Software Corp., (1999).

[9] Pearl, J., "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference", Morgan Kaufmann Publishers, Inc, (1988).

[10] Rasch, G., "Probabilistic Models for Some Intelligence and Attainment Tests", Danish Institute for Educational Research, (1960).

[11] Reye, J., "A belief Net Backbone for Student Modeling, Proceedings of Intelligent tutoring Systems", Third International Conference, ITS '96, Springer-Verlag, (1996).

[12] Ross, S., "A First Course in Probability", Prentice-Hall International, Inc., 5th edition, (1998).

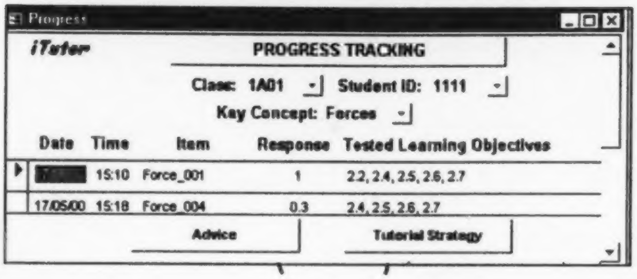
[13] Russell, S. & Norvig, P. , "Artificial Intelligence: A Modern Approach", Prentice Hall, (1995).

[14] Simon H.A., "The New Science of Management Decisions", New York: Harper & Row, (1960).

[15] TCI Software Research, "Scientific Notebook version 3.00 Build 939", Brooks / Cole Publishing Company, (1998).

[16] Villano, M., "Probabilistic Student Models: Bayesian Belief Networks and Knowledge Space Theory", Proceedings of Intelligent Tutoring Systems: Second International Conference, ITS '92, Springer-Verlag, (1992)

(a) User interface for teacher to track student's progress



Progress

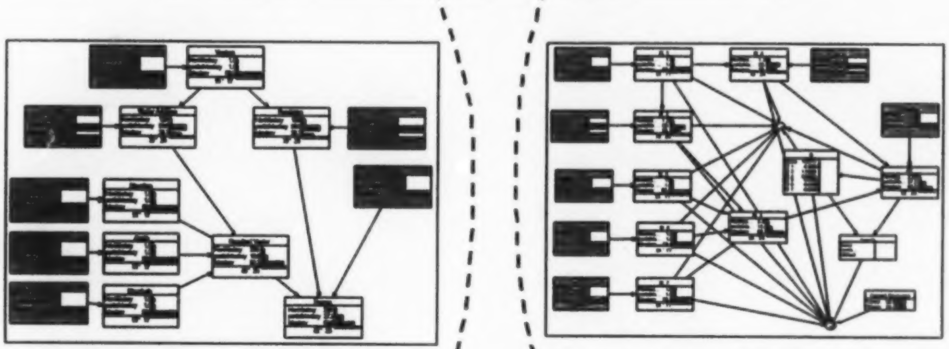
iTutor

PROGRESS TRACKING

Class: 1A01 Student ID: 1111

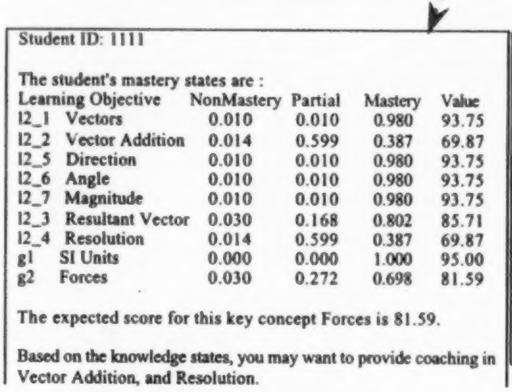
Key Concept: Forces

Date	Time	Item	Response	Tested Learning Objectives
[illegible]	15:10	Force_001	1	2.2, 2.4, 2.5, 2.6, 2.7
17/05/00	15:18	Force_004	0.3	2.4, 2.5, 2.6, 2.7



(b) Bayes's net running as background process (transparent to user)

(d) Dynamic decision network running as background process (transparent to user)



Student ID: 1111

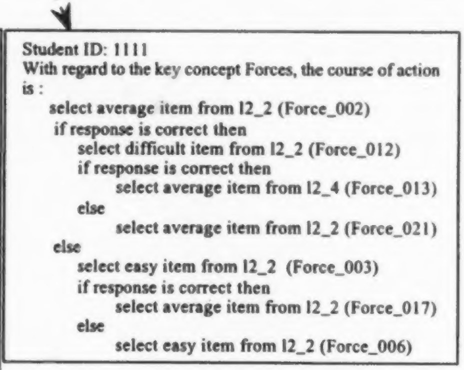
The student's mastery states are :

Learning Objective	NonMastery	Partial	Mastery	Value
I2_1 Vectors	0.010	0.010	0.980	93.75
I2_2 Vector Addition	0.014	0.599	0.387	69.87
I2_5 Direction	0.010	0.010	0.980	93.75
I2_6 Angle	0.010	0.010	0.980	93.75
I2_7 Magnitude	0.010	0.010	0.980	93.75
I2_3 Resultant Vector	0.030	0.168	0.802	85.71
I2_4 Resolution	0.014	0.599	0.387	69.87
g1 SI Units	0.000	0.000	1.000	95.00
g2 Forces	0.030	0.272	0.698	81.59

The expected score for this key concept Forces is 81.59.

Based on the knowledge states, you may want to provide coaching in Vector Addition, and Resolution.

(c) Output of student's mastery states



Student ID: 1111

With regard to the key concept Forces, the course of action is :

```
select average item from I2_2 (Force_002)
if response is correct then
    select difficult item from I2_2 (Force_012)
    if response is correct then
        select average item from I2_4 (Force_013)
    else
        select average item from I2_2 (Force_021)
else
    select easy item from I2_2 (Force_003)
    if response is correct then
        select average item from I2_2 (Force_017)
    else
        select easy item from I2_2 (Force_006)
```

(e) Output of tutoring strategy

Figure 9: Overview of an *iTutor* Session

BEST COPY AVAILABLE